Introduction to jQuery

5

LEARNING OBJECTIVES

After completing this chapter, you will be able to understand:

- jQuery configuration in your application.
- Basic syntax of jQuery.
- Various events and effects.
- Animated effects for HTML elements.
- How to manipulate DOM manipulation.

5.1 | Overview of jQuery

We had given a glimpse into jQuery already in Chapter 1. By now, you must be aware of JavaScript which is the most popular and the only scripting language for the front-end development. It is event-driven interpreted language used for manipulating the HTML DOM elements. jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It was designed to ease JavaScript programming in the front-end. It is classified as JavaScript's light-weight library and used in more than 50% of the top 10,000 websites in the world. jQuery's vision is to enable web designers to write fewer lines of code for more functionality. Before jQuery's emergence, many routine tasks required several lines of code in JavaScript. To address this, jQuery was introduced. With jQuery, those huge lines of code were encompassed into callable methods. This means that the code of jQuery is simply JavaScript – fortunately for us, someone simplified the most common JavaScript methods through jQuery.

The library can be used for HTML, CSS, DOM manipulation, to add effects and animations, to configure event methods in HTML, and to work with AJAX. Additionally, jQuery has a variety of plugins. However, before learning jQuery, you must have a beginner level knowledge of HTML, CSS, and JavaScript.

What is the difference between JavaScript and jQuery?

5.2 | Configuration of jQuery

There are two basic approaches for incorporating the jQuery library on your website.

Approach 1: You can download it from jQuery.com. The website offers two versions:

1. **Development Version:** This is used for development and testing purposes.
2. **Production Version:** This is used for live websites. It is different from the development version due to its state of being compressed and minified, that is, its code is optimized and minimized greatly for better performance.

The library is entailed in a single JavaScript file and it is specified in the <*script*> tag of the HTML. Since it is metadata for the HTML, hence it must be placed in the <*head*> tag.

```
<head>
<script src="jQuery-3.3.1.min.js"></script>
</head>
```

Note that the downloaded file has to be put in the directory which houses the other webpages. In case, you wish to avoid downloading jQuery, then you have a better option.

Flash Quiz What is the difference between library and framework?

Approach 2: Make use of a content delivery network (CDN) like Microsoft or Google. Most of the visitors have jQuery from Microsoft or Google when they go around websites. Therefore, it is called from the cache memory when visitors will interact with your websites. As a result, they get a quicker website loading time.

For Microsoft's jQuery integration, write the following code:

```
<head>
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jQuery-3.3.1.min.js"></script>
</head>
```

For Google's jQuery integration, write the following code:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</head>
```

When a website has several webpages, then web designers require some organization, especially if there are many jQuery functions. To do this, you can always store your functions in a different file – the file must have a *.js* extension. In this chapter, for simplicity, we are going to add them in the *<head>* tag.

Flash Quiz Is jQuery a library or a framework?

5.3 | Syntax

In jQuery, HTML elements are "selected" or queried so that an action can be performed using them, but wait! Have not we practiced a similar practice with CSS? So what is the point of jQuery? Well, first of all, since jQuery is all JavaScript, hence you have your grip on a real programming language to modify your front-end. For example, you can use for loop in the front-end with jQuery, but the same cannot be done with CSS. Likewise, jQuery assists to add interactivity which is not possible with CSS.

The standard format for a jQuery syntax is

```
$(selector).action()
```

By breaking down this format, we have three elements. First, we have the *$* sign that accesses jQuery. Second, we have a *selector* that searches for the specified HTML elements. Last, we have an *action* which executes a task on the selected elements. This syntax is not too dissimilar from CSS.

In our examples, we will only use methods which are defined by a document-ready event. The purpose of this approach is to block the execution of the jQuery code until the document is loaded or "*ready*". This wait is recommended because it allows the complete document to be shown before jQuery performs any task. For example, jQuery can attempt to hide an HTML element which is not loaded yet on the webpage. Likewise, it can alter an image's size before it is loaded. Additionally, this allows for JS code to be defined in the head tag, before the HTML body. Generally, this event looks like the following:

```
$(document).ready(function(){
// Any jQuery method will be defined here
});
Alternatively, there is a comparatively easier and more concise method:
$(function(){
// Any jQuery method will be defined here
});
```

Flash Quiz Will the page throw an error if we do not use *$(document).ready* function?

5.4 | Selectors

HTML elements are manipulated through selectors in jQuery. They "*select*" or search for elements via their *id*, *classes*, *types*, *attributes*, or simply by their name. Selectors begin with "$" sign and a set of parentheses "( )".

Flash Quiz Can you have more than one "document.ready" functions?

5.4.1 Element

To find an HTML element, the element selector is used. For instance, to select a paragraph tag <*p*>, you can write: *$("p")*. To understand this better, observe the following:

```
<!DOCTYPE html>
<html>
   <head>
      <script
         src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
      <script>
         $(document).ready(function() {
            $("button").click(function() {
               $("p1").hide();
            });
         });
      </script>
   </head>
   <body>
      <h2>This Example Demonstrates Element Selector</h2>
      <p1>The first paragraph is about to disappear by using jQuery. <br></br></p1>
      <p2>The second paragraph is not going to disappear.</p2>
      <button id="btn">Click Here to Disappear the First Paragraph</button>
   </body>
</html>
```

Let us go through this code to understand what happens at each stage. When we open this file in a browser, the browser renders the html code and creates a Document Object Model (DOM) tree. Once the DOM tree is created, browser then creates a Cascading Style Sheet Object Model (CSSOM) tree to map CSS with HTML elements. Since browser renders HTML code, as soon as it finds <*script*> tag, it starts executing it line by line. Some browsers work differently; for example, Mozilla Firefox downloads the scripts in the background so it continues with executing HTML code.

Let us dissect our above example and see how the mentioned jQuery code gets executed.

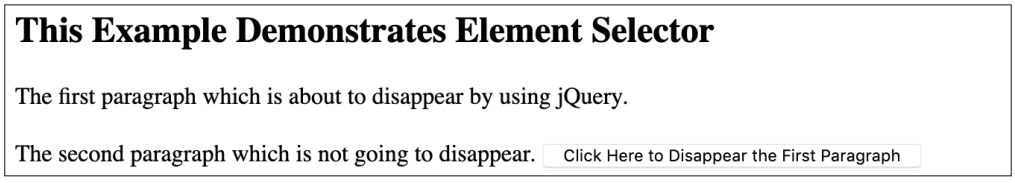
Step 1: First, we have added a *<script>* tag to specify the jQuery source file. The browser then downloads this file and executes it. This gives the ability to render jQuery code. The browser can now know the jQuery tags it is going to encounter.

Step 2: Then it reaches to `$(document).ready(function() {` line. This line tells the browser not to execute the code from this block until the document is fully loaded. In other words, page DOM is ready for jQuery to execute.

Step 3: Once the execution pointer is inside the ready function, it comes to the next line which is `$("button").click(function() {`. This code get executed when click event occurs on the button element.

Step 4: When it detects the "click" event, it enters into the `$("button").click(function() {` and finds `$("p1").hide();`. In this, we have used a selector on "p1" tag. So, jQuery uses this to find the p1 element and hide it.

Let us see the following output of the above code. This clearly shows how easy it is to select an element and perform an action on it. The following image is taken before pressing the button.



Once the button is pressed, the first paragraph is gone (see the following image).

Flash Quiz

Can you use your choice of character other than $ on jQuery elements?

5.4.2 #id Selector

Similarly, the *id* attribute of an HTML element can also be used by jQuery to configure functionality. Note that an *id* is always unique (in a single webpage). Hence, the id selector is needed to search for a unique element. To use this selector, you must use a hash character before the element *id*. As an example, see the following code:

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
        <script>
            $(document).ready(function() {
                $("button").click(function() {
                    $("#first").hide();
                });
            });
        </script>
    </head>
    <body>
        <h2>This Example Demonstrates ID Selector</h2>
        <p1 id="first">The first paragraph is about to disappear by using jQuery. <br>
</br></p1>
        <p2>The second paragraph which is not going to disappear.</p2>
        <button>Click Here to Disappear the First Paragraph</button>
    </body>
</html>
```

This code produces the following result which shows the page state before clicking the button:

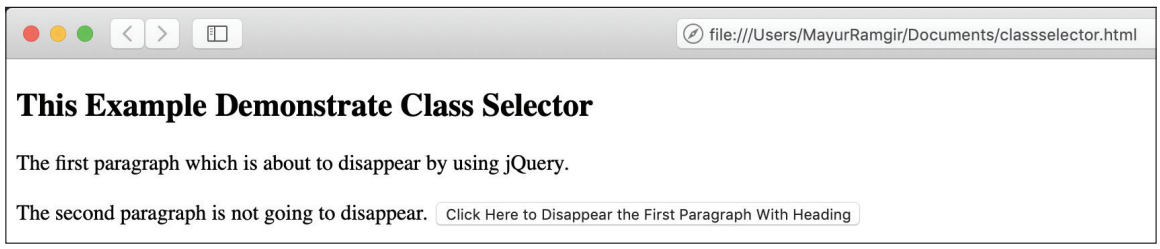
Once the button is clicked, the first paragraph disappears. See the following image for better understanding.

5.4.3 Class Selector

Likewise, you can use the *class* selector for choosing any HTML element with a defined class. To use this selector, you have to add a period character "." defining the name of the class. Do remember that the difference between an HTML *id* and *class* is that multiple elements can share the same class; hence it is used for grouping. For example,

```
<!DOCTYPE html>
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
        <script>
            $(document).ready(function() {
                $("button").click(function() {
                    $(".hd").hide();
                });
            });
        </script>
    </head>
    <body>
        <h2 class="hd">This Example Demonstrates Class Selector</h2>
        <p1 class="hd">The first paragraph is about to disappear by using jQuery. <br>
</br></p1>
        <p2>The second paragraph is not going to disappear.</p2>
        <button>Click Here to Disappear the First Paragraph With Heading</button>
    </body>
</html>
```

This code produces the following result. This page state is of before clicking the button.



After the button is clicked, the following result will be displayed. Here you can notice that the elements with the same class as "*hd*" have disappeared.

Flash Quiz

Can you unhide an element after using *.hide* on it?

5.5 | Events

jQuery is also used for manufacturing response to an "*event*" in a webpage. Whenever a visitor performs an action on an HTML page, there are pre-defined replies to provide them with a response. This pre-defined reply is called as an event. An event can be triggered when a user clicks on the *submit* button, or checks a radio-button, or even if they simply hover a mouse around an element. The triggering state of an event is often characterized by the term "*fire*".

5.5.1 Syntax

Usually, DOM events are aided by a jQuery method. To make sure that the <*h*1> heading is clicked to trigger an event, you can write the following:

```
$("h1").click().
```

Afterwards, you have to specify the functionality so that a reply could be integrated whenever the event is triggered.

5.5.2 jQuery Event Methods

As we have seen, event is like a response which is triggered by a user action. jQuery provides methods to capture these user actions and reply to them.

5.5.2.1 `click()`

As the name suggests, this function is executed whenever an HTML element is clicked by a user. It fixes an event handler function with an HTML element. In the following example, the click method is assigned to HTML headings. When a user clicks on them, the heading gets hidden. This happens because the click event is registered for the elements used as the selector like `$("h1, h2, h3").click(function(){`. This tells jQuery to execute the code in this block when a click event occurs on h1, h2, or h3 elements.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h1, h2, h3").click(function(){
        $(this).hide();
     });
    });
    </script>
</head>
<body>
    <h1></h1>
    <h1>First Heading</h1>
    <h2>Second Heading</h2>
    <h3>Third Heading</h3>
</body>
</html>
```

This code shows the following result in the browser window.

After clicking the button for the first time, the following result will be displayed.

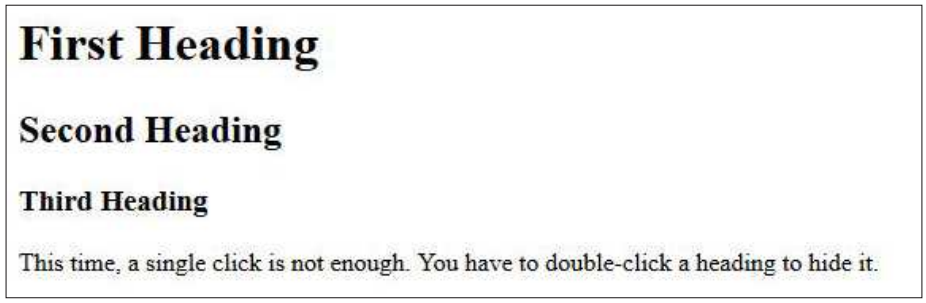
After clicking the button the second time, the following result will be displayed.

5.5.2.2 dblclick()

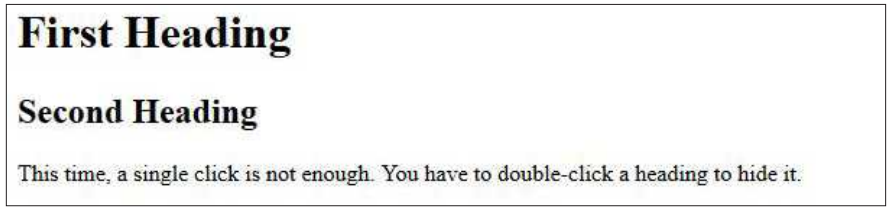
To fix an event handler function with an HTML element, the `dblclick()` function is used. Let us change the example discussed in the previous subsection. In the following example, our headings are only going to be removed when the user double-clicks them.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h1,h2,h3").dblclick(function(){
        $(this).hide();
     });
    });
    </script>
</head>
    <body>
    <h1>First Heading</h1>
    <h2>Second Heading</h2>
    <h3>Third Heading</h3>
    <p>This time, a single click is not enough. You have to double-click a heading to
hide it.</p>
</body>
</html>
```

This code shows the following result in the browser window.



After the button click, the following result will be displayed.

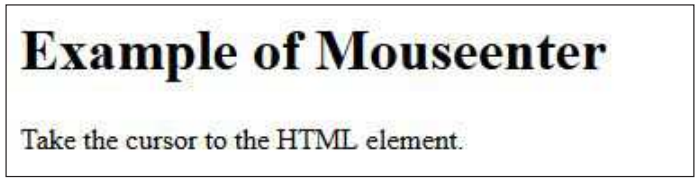


5.5.2.3 `mouseenter()`

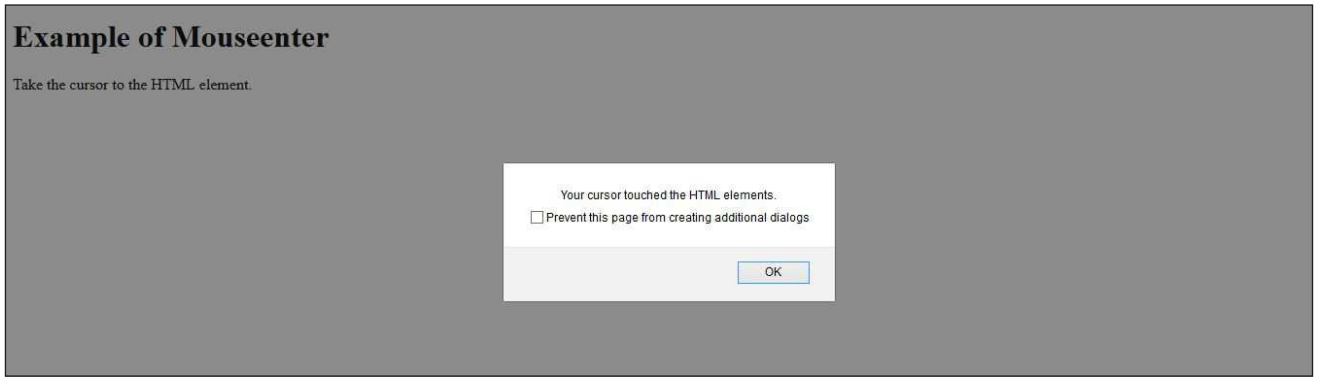
Similarly, the `mouseenter()` method is used for integrating an event handling functionality with an HTML element. Check this example, where whenever the mouse cursor enters the area of the HTML elements, an alert notifies the user.

```
<!DOCTYPE html>
<html>
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
   <script>
   $(document).ready(function(){
    $(".a").mouseenter(function(){
       alert("Your cursor touched the HTML elements.");
    });
   });
   </script>
</head>
<body>
   <h1 class="a">Example of Mouseenter</h1>
   <p class="a">Take the cursor to the HTML element.</p>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the page appears as follows.

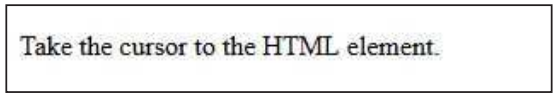


5.5.2.4 mouseleave()

The `mouseleave()` function is triggered when the cursor of a user leaves an HTML element.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#par").mouseleave(function(){
        alert("You left the paragraph element!");
     });
    });
    </script>
</head>
<body>
    <p id="par">Take the cursor to the HTML element.</p>
</body>
</html>
```

This code shows the following result in the browser window:



After the button click, the following result will be displayed.

5.6 | Effects

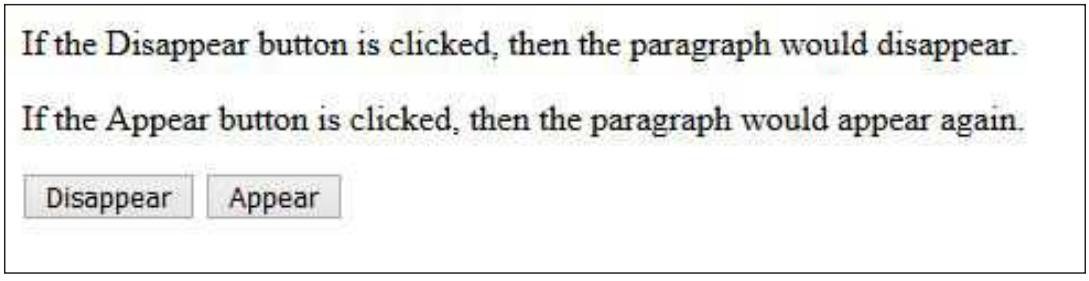
With jQuery, a whole variety of effects can be generated on the webpages. To enhance the user experience, we can consider adding some visual effects for actions such as button click, popup window open and close, selecting a dropdown value, show or hide various elements such as text boxes, etc.

5.6.1 `hide()` and `show()`

In jQuery, HTML elements can appear and disappear by using the `hide()` and `show()` methods. In our previous examples, we used `hide()` method to make some of the HTML elements disappear. However, we have not yet seen the `show()` method. Let us check what happens when both are used at a time:

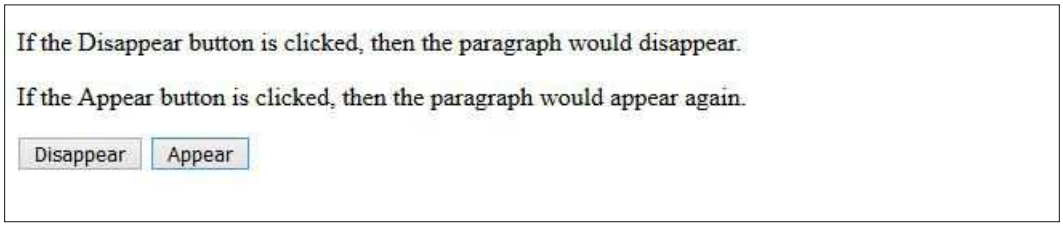
```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#a").click(function(){ //This captures the click event on element with id "a"
        $("p").hide(); //This hides all the elements p
     });
     $("#b").click(function(){//This captures the click event on element with id "b"
        $("p").show(); //This shows all the elements p
     });
    });
    </script>
</head>
    <body>
    <p>If the Disappear button is clicked, then the paragraph would disappear.</p>
    <p>If the Appear button is clicked, then the paragraph would appear again.</p>
    <button id="a">Disappear</button>
    <button id="b">Appear</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the *Disappear* button is clicked, the following result will be displayed.

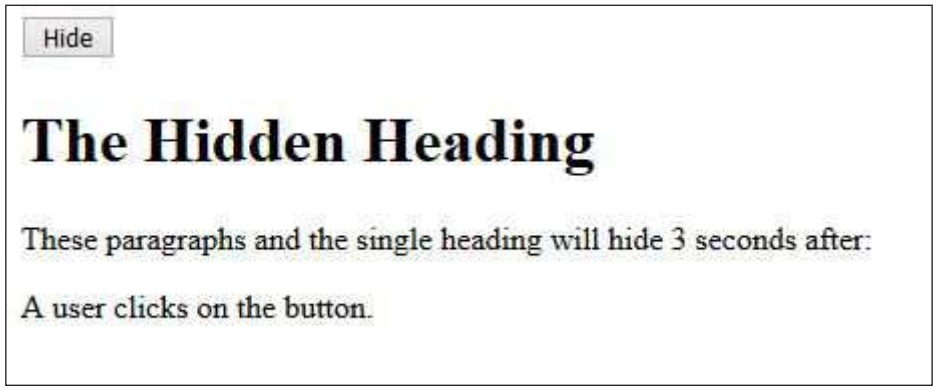
After the *Appear* button is click, the following result will be displayed.



The parentheses of both `hide()` and `show()` function can take two optional arguments: *speed* and *callback*. Speed refers to the time for hiding or showing. It can be defined by "fast", "slow", or milliseconds (digits). On the other hand, the callback argument performs its task when the show or hide method finishes. For instance, to make sure that your HTML elements disappear after 3 seconds, you can write the following piece of code:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $(".a").hide(3000);
     });
    });
    </script>
</head>
<body>
    <button>Hide</button>
    <h1 class ="a">The Hidden Heading </h1>
    <p class="a">These paragraphs and the single heading will hide 3 seconds after:</p>
    <p class="a">A user clicks on the button. </p>
</body>
</html>
```

This code shows the following result in the browser window.



After the button click, the following result will be displayed.

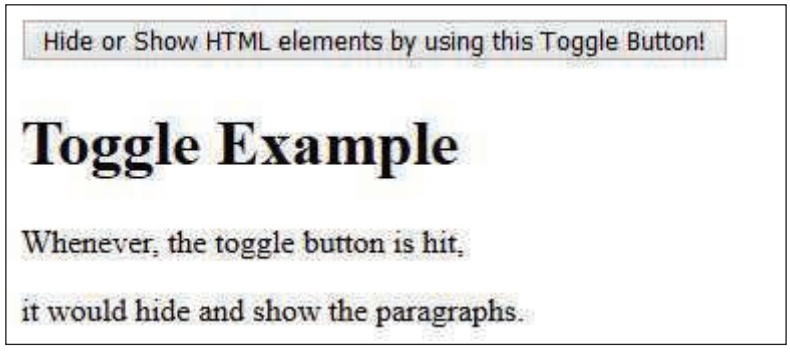
QUICK CHALLENGE Write a code to hide and show an element in a periodic interval without any action from the user.

5.6.2 `toggle()`

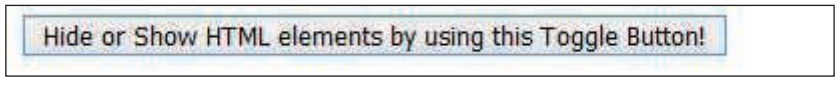
To turn on or turn off a functionality like showing or hiding HTML elements, you can use the *toggle()* method. For example,

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $(".a").toggle();
     });
    });
    </script>
</head>
<body>
    <button>Hide or Show HTML elements by using this Toggle Button!</button>
    <h1 class ="a">Toggle Example </h1>
    <p class="a">Whenever, the toggle button is hit,</p>
    <p class="a">it would hide and show the paragraphs. </p>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.



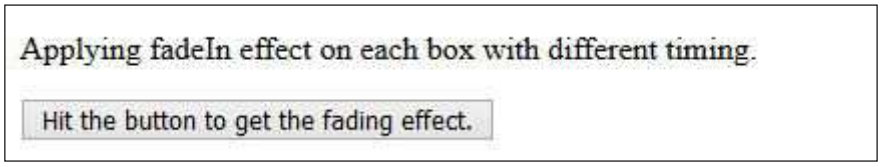
5.6.3 Fading

For incorporating fading with an element, jQuery provides support with four methods: `fadeIn()`, `fadeout()`, `fadeToggle()`, `fadeTo()`.

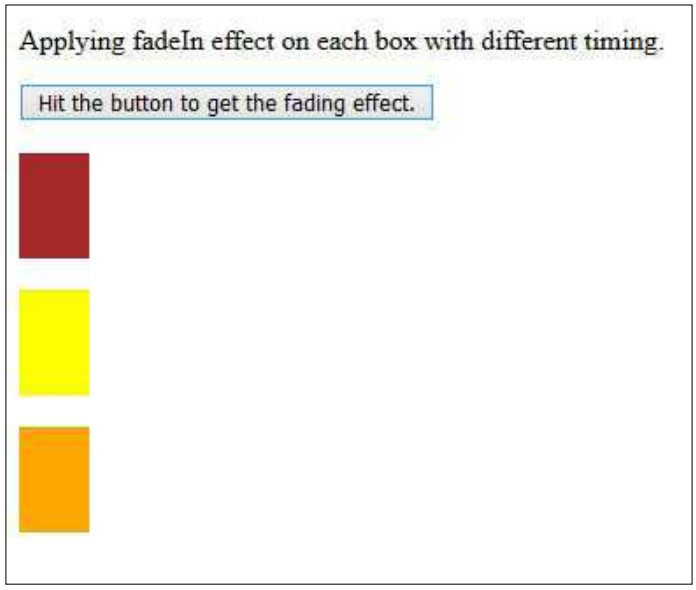
`fadeIn()`: This method is useful to change the opacity of an element from hidden to visible. Let us see the following example with `fadeIn()`:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("#1").fadeIn(1000);
        $("#2").fadeIn(2000);
        $("#3").fadeIn(3000);
     });
    });
    </script>
</head>
<body>
    <p>Applying fadeIn effect on each box with different timing.</p>
    <button>Hit the button to get the fading effect.</button><br><br>
    <div id="1" style="width:40px;height:60px;display:none;background-color:brown;">
</div><br>
    <div id="2" style="width:40px;height:60px;display:none;background-color:yellow;">
</div><br>
    <div id="3" style="width:40px;height:60px;display:none;background-color:orange;">
</div>
</body>
</html>
```

This code shows the following result in the browser window.



After the button click, the following result will be displayed.



QUICK CHALLENGE Define `fadeout()`, `fadeToggle()`, and `fadeTo()` in detail. Give examples of each.

5.6.4 Sliding

jQuery is often known for its beautiful and interactive sliders. It offers three basic slide methods: `slideDown()`, `slideUp()`, and `slideToggle()`.

1. **`slideDown()`:** This method is useful to reveal an element slowly, from top to bottom. It animates the height of the element to reveal the lower part of the element.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#flip").click(function(){
        $("#panel").slideDown("fast");
     });
    });
    </script>
    <style>
    #panel, #flip {
     padding: 10px;
     text-align: left;
     background-color: dodgerblue;
     border: solid 5px purple;
    }
    #panel {
     padding: 60px;
     display: none;
    }
    </style>
</head>
<body>
    <div id="flip">When this area is clicked, a panel emerges.</div>
    <div id="panel">The panel greets the user with Merry Christmas!</div>
</body>
</html>
```

This code shows the following result in the browser window.

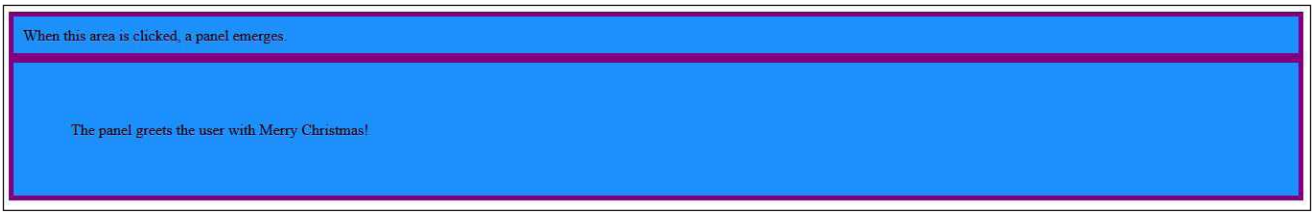
After the button is clicked, the following result will be displayed.

Similarly, to slide up an HTML element, we can use the `slideUp()` method.

2. **`slideUp():`** This method is useful to sliding up an element from bottom to top, which gives animated effect of concealing. We show this using the example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#flip").click(function(){
        $("#panel").slideUp("fast");
     });
    });
    </script>
    <style>
    #panel, #flip {
     padding: 10px;
     text-align: left;
     background-color: dodgerblue;
     border: solid 5px purple;
    }
    #panel {
     padding: 60px;
    }
    </style>
</head>
<body>
    <div id="flip">When this area is clicked, a panel emerges.</div>
    <div id="panel">The panel greets the user with Merry Christmas!</div>
</body>
</html>
```

This code shows the following result in the browser window.



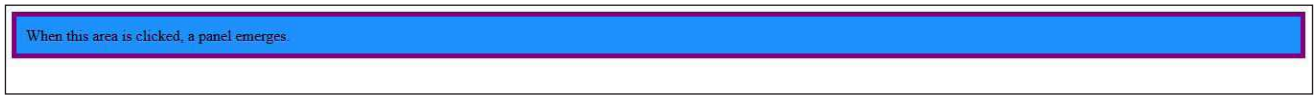
After the button is clicked, the following result will be displayed.

5.6.4.1 `slideToggle()`

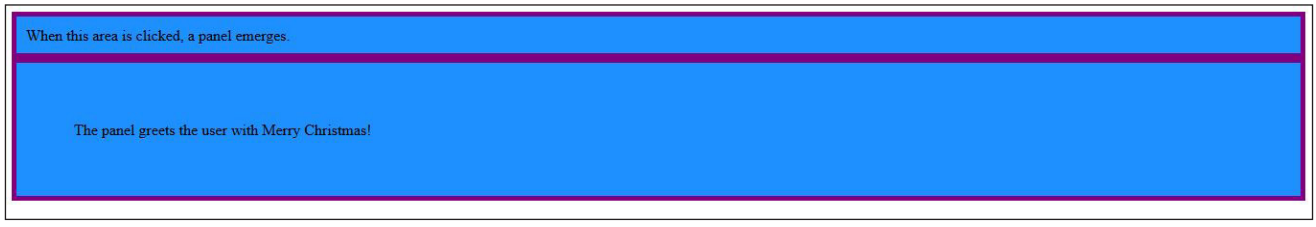
To enable or disable sliding, you can use the `slideToggle()` method. This method is useful to enable or disable sliding. In other words, if the element at its full height then it conceal it, and if it is concealed then it reveals the element. Let us see the example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#flip").click(function(){
        $("#panel").slideToggle("fast");
     });
    });
    </script>
    <style>
    #panel, #flip {
     padding: 10px;
     text-align: left;
     background-color: dodgerblue;
     border: solid 5px purple;
    }
    #panel {
     padding: 60px;
     display: none;
    }
    </style>
</head>
<body>
    <div id="flip">When this area is clicked, a panel emerges.</div>
    <div id="panel">The panel greets the user with Merry Christmas!</div>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.



5.6.5 Animation

It is also possible to add movements via animations in jQuery. To do this, you require the `animate()` method. The basic syntax of the `animate()` method is the following:

```
$(selector).animate({params}, speed, callback);
```

Params is short for parameter and references to CSS properties that require animation. Speed refers to the time period for animation – "fast", "slow", and milliseconds are arguments. The callback parameter runs after the completion of the animation. We would read more on *Callback* in a later section.

Note that by default, HTML elements are static which means it is not possible to move them. To move them, you have to adjust the CSS property of that element to absolute, fixed, or relative.

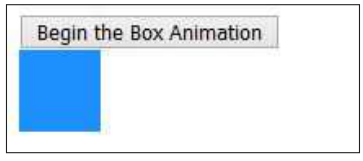
For a simple animation, check the following example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("div").animate({left: '500px'});
     });
    });
    </script>
</head>
<body>
    <button>Move the box!</button>
    <p></p>
    <div style="background:dodgerblue;height:50px;width:50px;position:fixed;"></div>
</body>
</html>
```

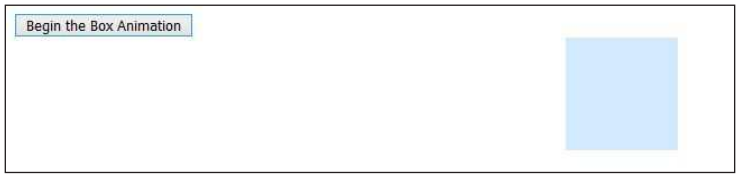
Let us move towards more advanced animations where we add multiple properties to our box.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("div").animate({
         left: '500px',
         opacity: '0.2',
         height: '100px',
         width: '100px'
        });
     });
    });
    </script>
</head>
<body>
    <button>Begin the Box Animation</button>
     <div style="background:dodgerblue;height:50px;width:50px;position:fixed;"></div>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.

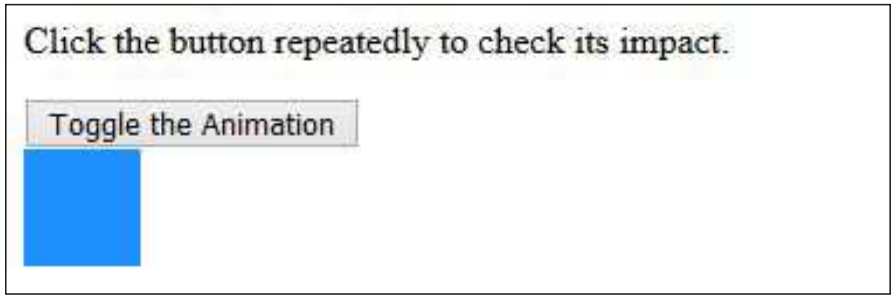


You can also use the *hide*, *show*, and *toggle* values in the animate method. For instance, consider the following example:

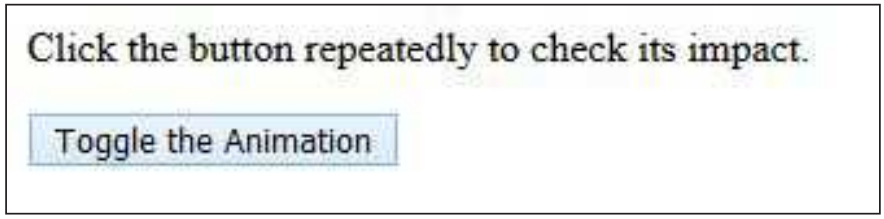
```
<!DOCTYPE html>
<html>
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
   <script>
   $(document).ready(function(){
    $("button").click(function(){
       $("div").animate({
        width: 'toggle'

       });
    });
   });
   </script>
</head>
<body>
<p>Click the button repeatedly to check its impact.</p>
   <button>Toggle the Animation</button>
   <div style="background:dodgerblue;height:50px;width:50px;position:absolute;">
   </div>
</body>
</html>
```

This code shows the following result in the browser window.



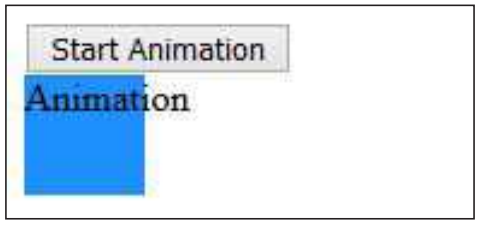
After the button is clicked, the following result will be displayed.



jQuery supports queue functionality. Hence, if multiple calls are written in the `animate()` method, then jQuery builds a queue with them, that is, calls animate step-by-step like a real-world queue. For example

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        var div = $("div");
        div.animate({height: '50px', opacity: '0.3'}, "slow");
        div.animate({width: '300px', opacity: '0.7'}, "fast");
        div.animate({height: '50px', opacity: '0.3'}, "fast");
        div.animate({width: '300px', opacity: '0.7'}, "slow");
            div.animate({fontSize: '2em'}, "slow");
     });
    });
    </script>
</head>
<body>
    <button>Start Animation</button>
    <div style="background:dodgerblue;height:50px;width:50px;position:absolute;">
Animation</div>
</body></html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.

Note that this time we also added text in our animation and used it in the queue to animate it as well.

5.6.6 jQuery Stop Animations

When an animation or effect is still in process, then the `stop()` method can halt it before its completion. This method is applicable to *fading*, *sliding*, *custom animations*, and any effect functions. The basic format of the method is as follows:

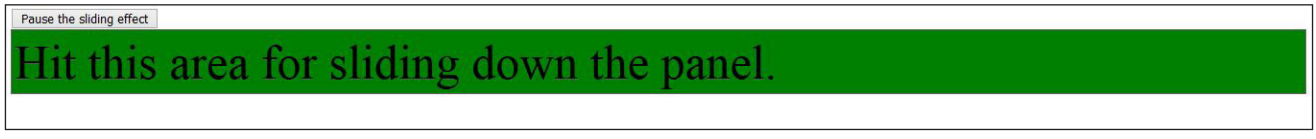
```
$(selector).stop(stopAll, goToEnd);
```

These parameters are optional. The "*stopAll*" parameter is used to specify if the queue in the animation requires to be cleared. By default, it is *false*; therefore, only the animation which is active is blocked, thereby permitting other animations in the queue to be executed. The "*goToEnd*" parameter defines if the current animation should be finished quickly. By default, it is *false*.

The `stop()` method stops any current animation, depending upon the element which is selected. For instance, consider the following example.

```
<!DOCTYPE html>
<html>
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
   <script>
   $(document).ready(function(){
    $("#flip").click(function(){
       $("#panel").slideDown(2000);
    });
    $("#stop").click(function(){
       $("#panel").stop();
    });
   });
   </script>
   <style>
   #panel, #flip {
    padding: 3px;
    font-size: 50px;
    text-align: left;
    background-color: green;
    color: black;
    border: solid 1px #666;
   }
   #panel {
    padding: 20px;
    display: none;
   }
   </style>
</head>
<body>
   <button id="stop">Pause the sliding effect</button>
   <div id="flip">Hit this area for sliding down the panel.</div>
   <div id="panel">Hello User! </div>
</body>
</html>
```

This code shows the following result in the browser window.



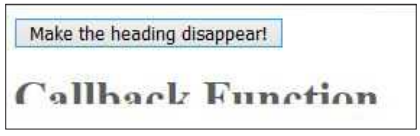
5.6.7 Callback Function

By default, JavaScript functions are executed and processed on a line-by-line basis. Sometimes, this is problematic as the next line of code is executed before the completion of an effect. As a consequence, your webpage may face errors. In order to avoid this, we use a callback function. When the current effect is completed, then the callback function is executed. For instance, consider the following example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("h1").hide(3000, function(){
         alert("The heading has disappeared.");
        });
     });
    });
    </script>
</head>
<body>
    <button>Make the heading disappear!</button>
    <h1>Callback Function</h1>
</body></html>
```

This code shows the following result in the browser window.

After the button click, the following result will be displayed.

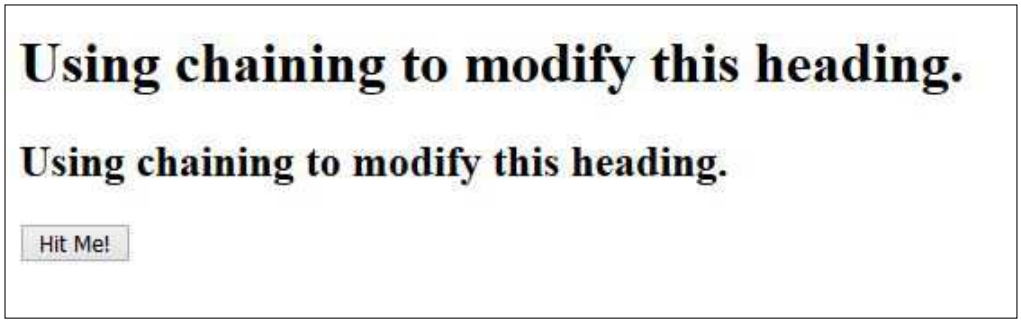


5.6.8 Chaining

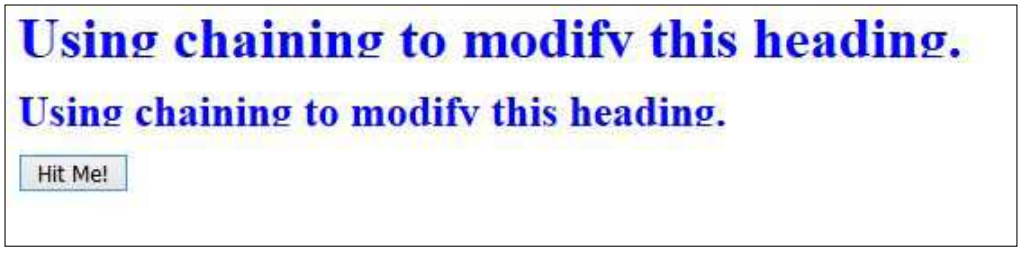
As the name suggests, chaining is the "chain" of multiple jQuery methods. Chaining in jQuery refers to the execution of more than a single jQuery method with a single element in one statement. This is useful for browsers because they do not have to search for an element repeatedly. For chaining, you just have to append the prior action to the current action. For instance,

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $(".h").css("color", "blue").slideUp(2000).slideDown(2000);
     });
    });
    </script>
</head>
<body>
    <h1 class="h">Using chaining to modify this heading.</h1>
    <h2 class="h">Using chaining to modify this heading.</h2>
    <button>Hit Me!</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.



5.7 | Working with HTML

jQuery contains powerful methods for changing and manipulating HTML elements and attributes. In this section, we will cover various such methods and explain with examples the use of those.

5.7.1 DOM Manipulation

jQuery allows users to modify Document Object Model (DOM). DOM is a standard which is used to access HTML and XML documents. Properties and methods define DOM's programming interface. Property is something the element has and method is something the element can do. There are various DOM properties around but following are some of the more significant ones. Let us say "n" is a node object.

1. **n.innerHTML:** This property refers to the inner text value of an HTML element "n". However, note that it parses content as HTML so it takes longer to get value compared to nodeValue property. For example, say "n" refers to <a> element like Mayur Ramgir. In this case, n.innerHTML will give "Mayur Ramgir".
2. **n.nodeName:** This is very straightforward property as it gives the name of the node "n".
3. **n.nodeValue:** This property is similar to innerHTML in a sense as it gives the inner text of node "n". However, a major difference is it does not parse HTML and gets the value in text format. Hence, it is faster than innerHTML.
4. **n.parentNode:** This property gives access to the parent node of "n".
5. **n.childNodes:** This property gives access to the child node of "n".
6. **n.attributes:** This property gives all the attributes of node "n".

5.7.2 Get Content and Attributes

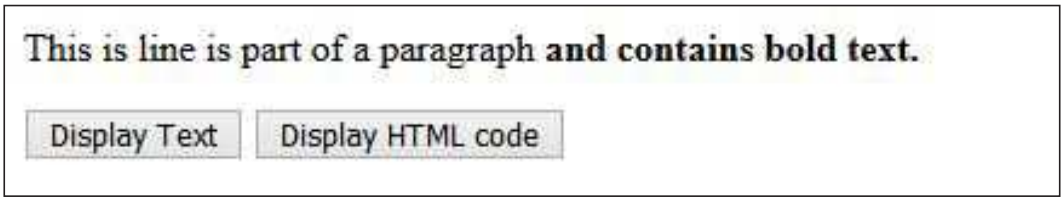
The following are three basic and easy-to-use methods in jQuery for DOM manipulation:

1. The `text()` method is used to assign or return any piece of text for the selected HTML elements.
2. The `html()` method is used to assign or return the content for the selected elements (markup included).
3. The `val()` method is used to assign or return any value from the fields in the HTML forms.

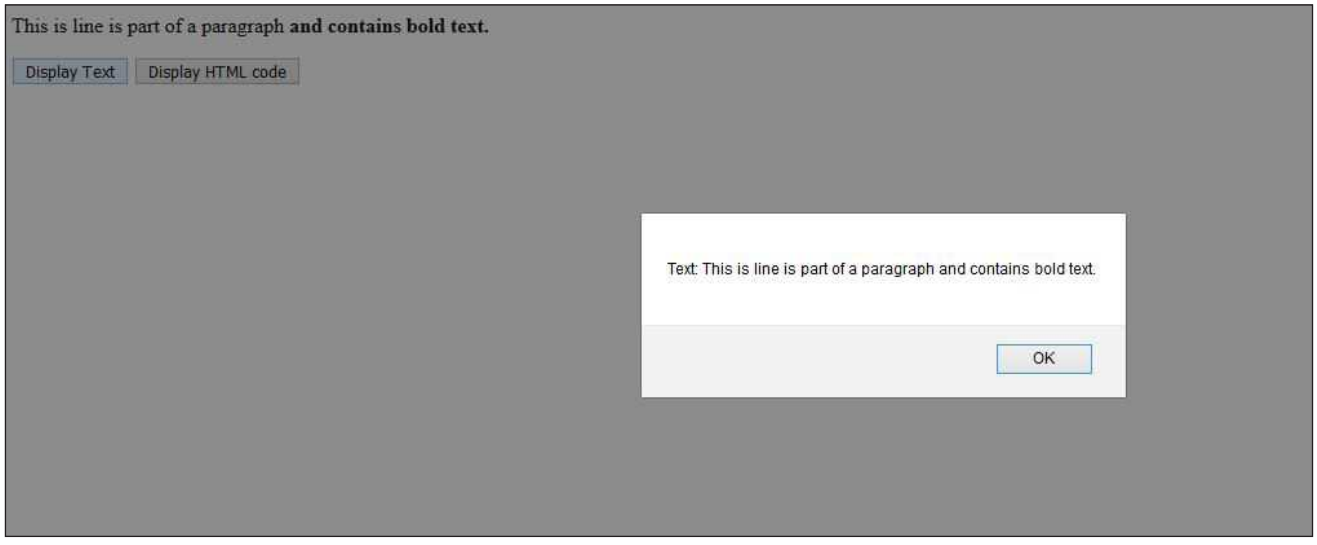
To see the `text()` and `html()` methods, consider the following example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#b1").click(function(){
        alert("Text: " + $("#par").text());
     });
     $("#b2").click(function(){
        alert("HTML: " + $("#par").html());
     });
    });
    </script>
</head>
<body>
    <p id="par">This is line is part of a paragraph <b>and contains bold text.</b> </p>
    <button id="b1">Display Text</button>
    <button id="b2">Display HTML code</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.

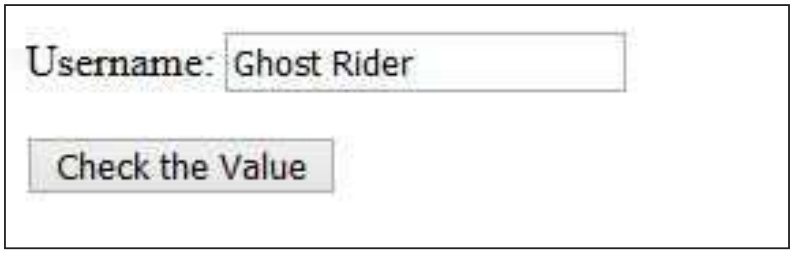


After the button is clicked again, the following result will be displayed.

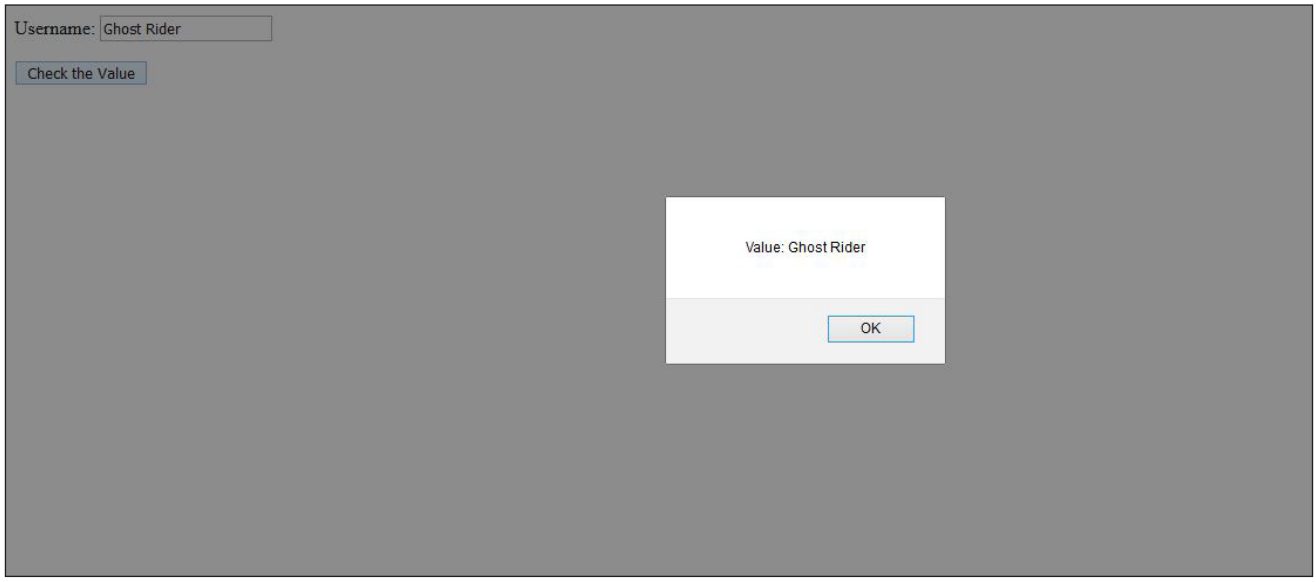
Similarly, to check the use of `val()`, consider the following:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#button1").click(function(){
        alert("Value: " + $("#txt").val());
     });
    });
    </script>
</head>
<body>
    <p>Username: <input type="text" id="txt" value="Ghost Rider"></p>
    <button id ="button1">Check the Value</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.



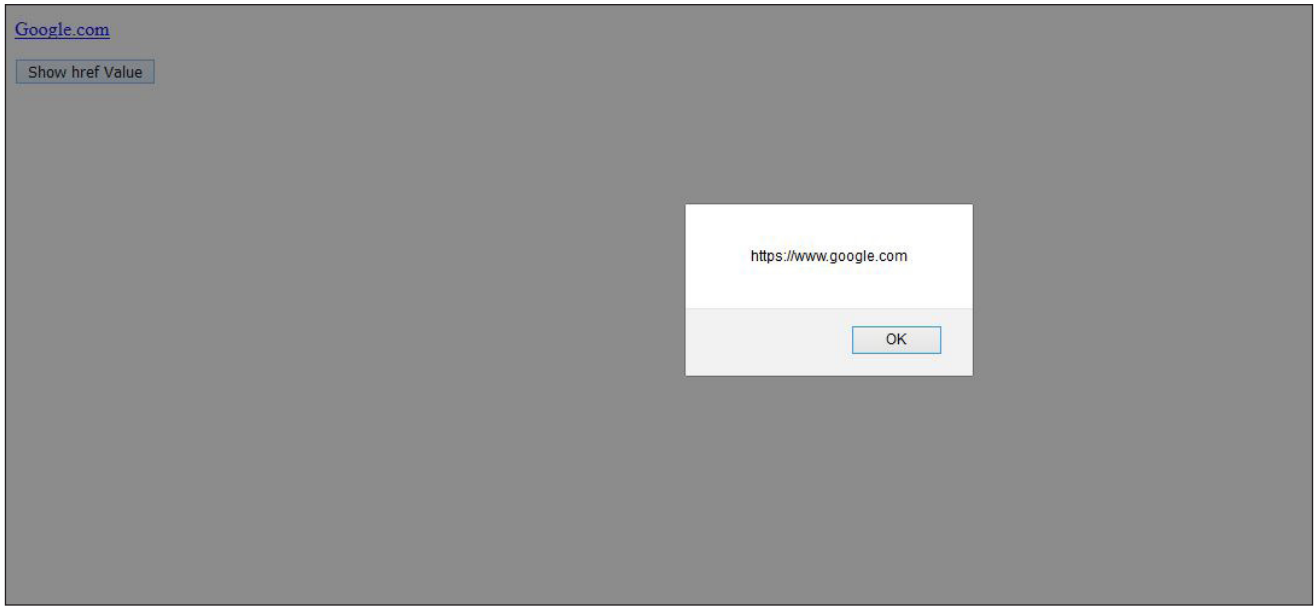
5.7.2.1 Get Attributes

To receive the value of HTML attributes, jQuery provides a method, `attr()`. Go through the following example in which the value of the *href* is displayed by the jQuery function.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        alert($("#go").attr("href"));
     });
    });
    </script>
</head>
<body>
    <p><a href="https://www.google.com" id="go">Google.com</a></p>
    <button>Show href Value</button>
</body>
</html>
```

This code shows the following result in the browser window.

After the button is clicked, the following result will be displayed.



5.7.3 Set Content and Attributes

The above-mentioned methods can also be used for setting content. For instance, check the following example in which we have used set content with the jQuery `html()`, `val()`, and `text()` methods.

```
<!DOCTYPE html>
<html>
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
   <script>
   $(document).ready(function(){
    $("#b1").click(function(){
       $("#par1").text("Hi User!");
    });
    $("#b2").click(function(){
       $("#par2").html("<b>Hi User!</b>");
    });
    $("#b3").click(function(){
       $("#par3").val("Shane");
    });
   });
   </script>
</head>
<body>
   <p id="par1">This is our first paragraph.</p>
   <p id="par2">This is our second paragraph.</p>
   <p>Input field: <input type="text" id="par3" value="Rick"></p>
   <button id="b1">Setting any textual information</button>
   <button id="b2">Setting any HTML data</button>
   <button id="b3">Setting any value</button>
</body>
</html>
```

This code shows the following result in the browser window.

After the button is clicked, the following result will be displayed.

After the button is clicked again, the following result will be displayed.

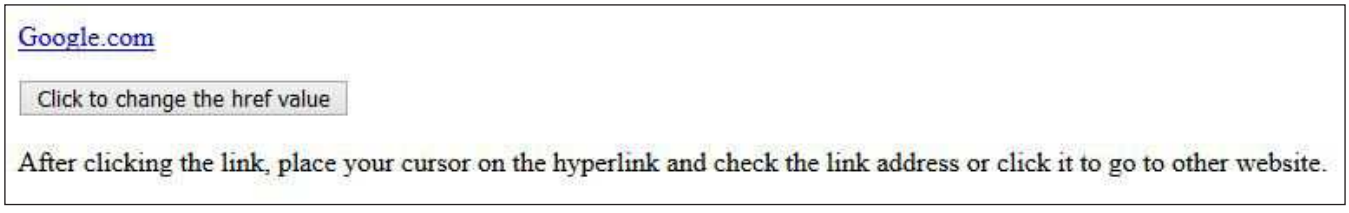
After the button is clicked once again, the following result will be displayed.

5.7.3.1 Set Attributes

To change or modify the values of an attribute, we can use the `attr()` method. Check the following example in which we have modified the value of *href* to redirect a user to another website. By now, you must have realized how jQuery can help you to gain a wider range of control and manipulation on the front-end coding of HTML.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#b1").click(function(){
        $("#w3s").attr("href", "https://www.reddit.com");
     });
    });
    </script>
</head>
<body>
    <p id="par1"><a href="https://www.google.com" id="w3s">Google.com</a></p>
    <button id="b1">Click to change the href value</button>
    <p>After clicking the link, place your cursor on the hyperlink and check the link
address or click it to go to other website.</p>
</body>
</html>
```

This code shows the following result in the browser window.



5.7.4 Addition of Elements

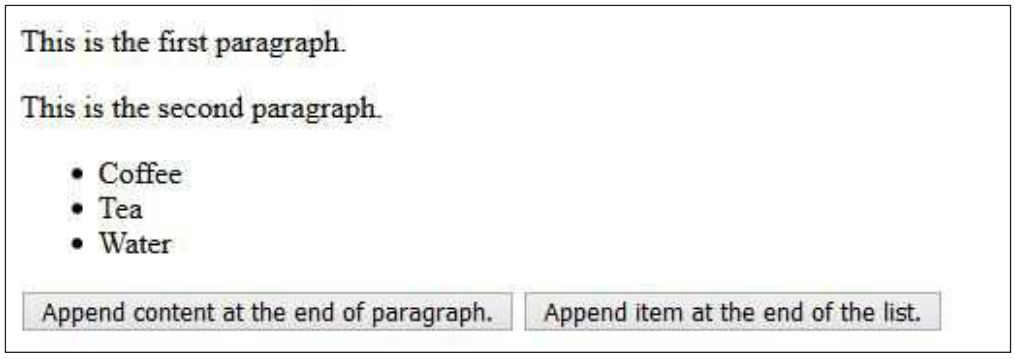
For the addition of fresh contents in HTML elements, jQuery provides four methods: `append()`, `prepend()`, `after()`, and `before()`.

5.7.4.1 append()

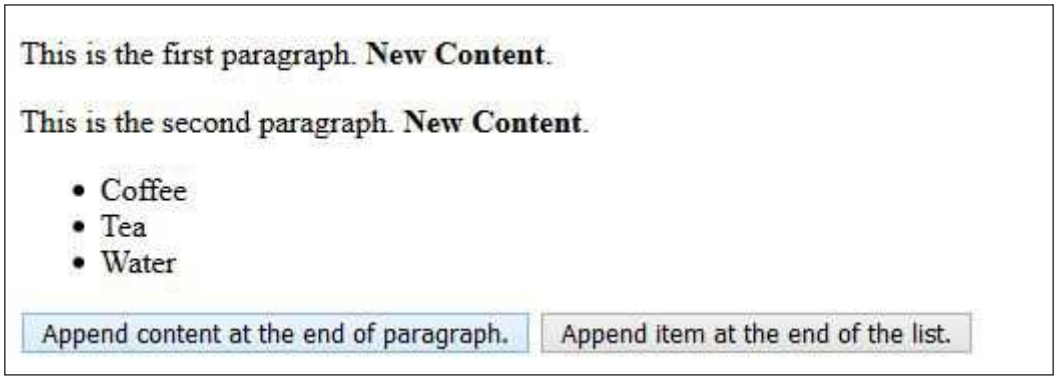
Whenever you want to insert any piece of content at the extreme end of your element, use the `append()` method. In the following example, we have appended text at the end of the paragraphs as well attached one more list item to the list.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#b1").click(function(){
        $("p").append(" <b>New Content</b>.");
     });
     $("#b2").click(function(){
        $("ul").append("<li>Soup</li>");
     });
    });
    </script>
</head>
<body>
    <p>This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
    <ul>
     <li>Coffee</li>
     <li>Tea</li>
     <li>Water</li>
    </ul>
    <button id="b1">Append content at the end of paragraph.</button>
    <button id="b2">Append item at the end of the list.</button>
</body>
</html>
```

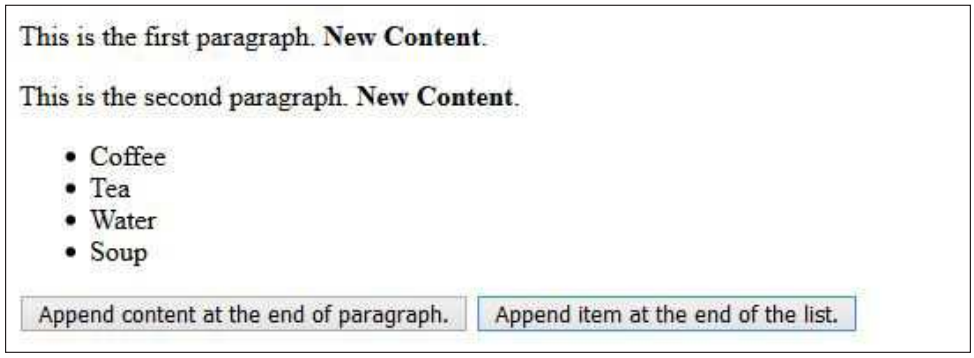
This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.



After the button is clicked again, the following result will be displayed.

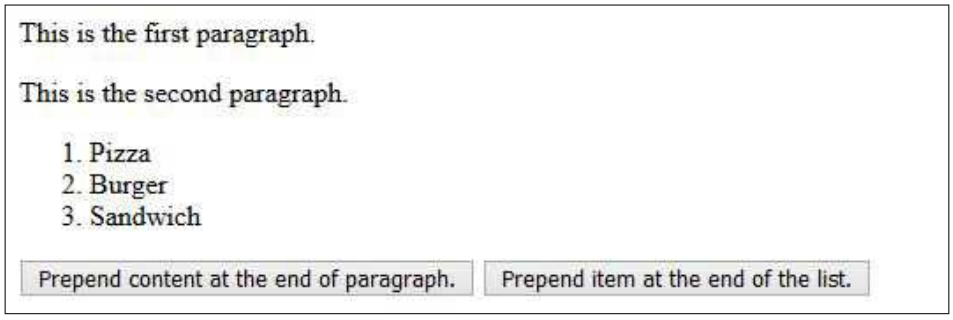


5.7.4.2 prepend()

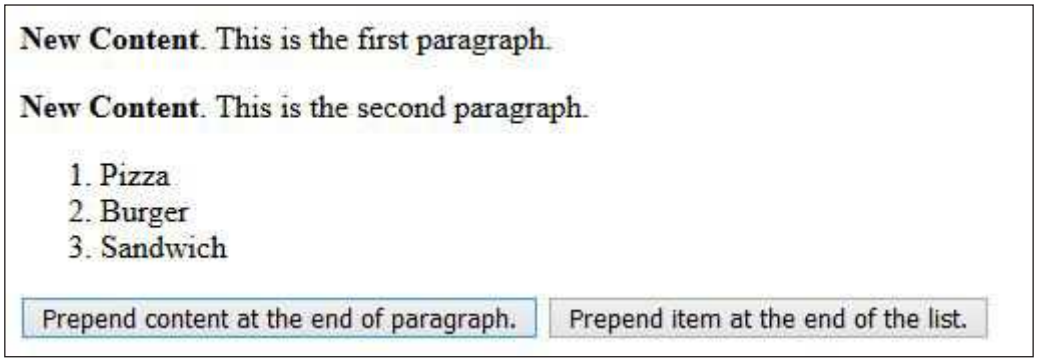
Similarly, in order to add content at the starting of an element, we can use the `prepend()` method.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#b1").click(function(){
        $("p").prepend("<b>New Content</b>. ");
     });
     $("#b2").click(function(){
        $("ol").prepend("<li>Nuggets</li>");
     });
    });
    </script>
</head>
<body>
    <p>This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
    <ol>
     <li>Pizza</li>
     <li>Burger</li>
     <li>Sandwich</li>
    </ol>
    <button id="b1">Prepend content at the end of paragraph.</button>
    <button id="b2">Prepend item at the end of the list.</button>
</body>
</html>
```

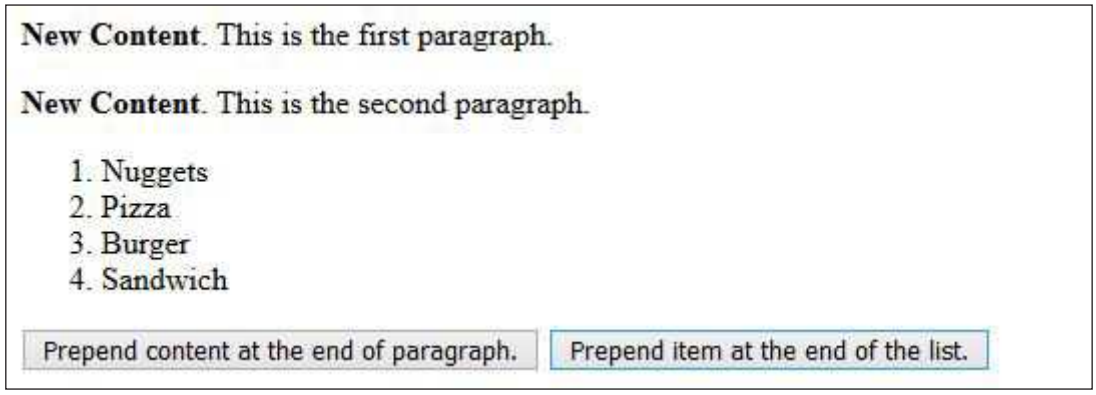
This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.



After the button is clicked again, the following result will be displayed.

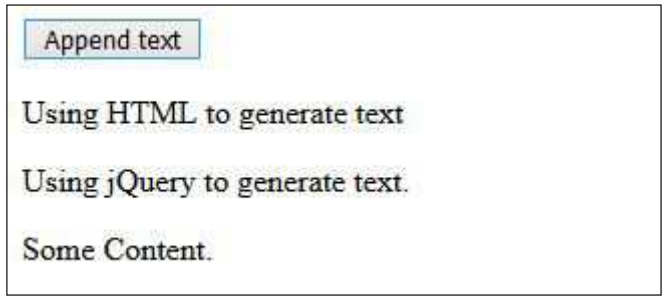


So far, we have written some basic examples with `append()` and `prepend()` but both of these methods are powerful enough to take elements as parameters. New HTML elements such as headings can be created from scratch by making use of jQuery. For instance, in the following example we have generated elements by using jQuery, HTML, and DOM. The `append()` method is used to add new elements.

```
<!DOCTYPE html>
<html>
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
   <script>
   function appTxt() {
    var t1 = "<p>Using HTML to generate text</p>";
    var t2 = $("<p></p>").text("Using jQuery to generate text.");
    var t3 = document.createElement("p");
    t3.innerHTML = "Some Content.";
    $("body").append(t1, t2, t3);
   }
   </script>
</head>
<body>
   <button onclick="appTxt()">Append text</button>
</body>
</html>
```

This code shows the following result in the browser window.

After the button is clicked, the following result will be displayed.

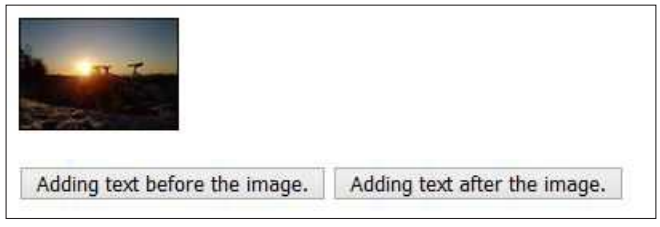


5.7.4.3 `after()` and `before()`

The `after()` and `before()` methods are used to add information after or before the specified elements. For example, to add content before and after an image, we can write the following code:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#b1").click(function(){
        $("img").before("<b>Adding text before the image.</b>");
     });
     $("#b2").click(function(){
        $("img").after("Adding text after the image.");
     });
    });
    </script>
</head>
<body>
    <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSiyG55GLGPMUDWyn_
luUeLq7XUny12hspiIL_9-3GEqWVWwNqQ" alt="jQuery" width="100" height="70">
    <br><br>
    <button id="b1">Adding text before the image.</button>
    <button id="b2">Adding text after the image.</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the button is clicked, the following result will be displayed.

After the button is clicked again, the following result will be displayed.

5.7.5 Deleting HTML Elements

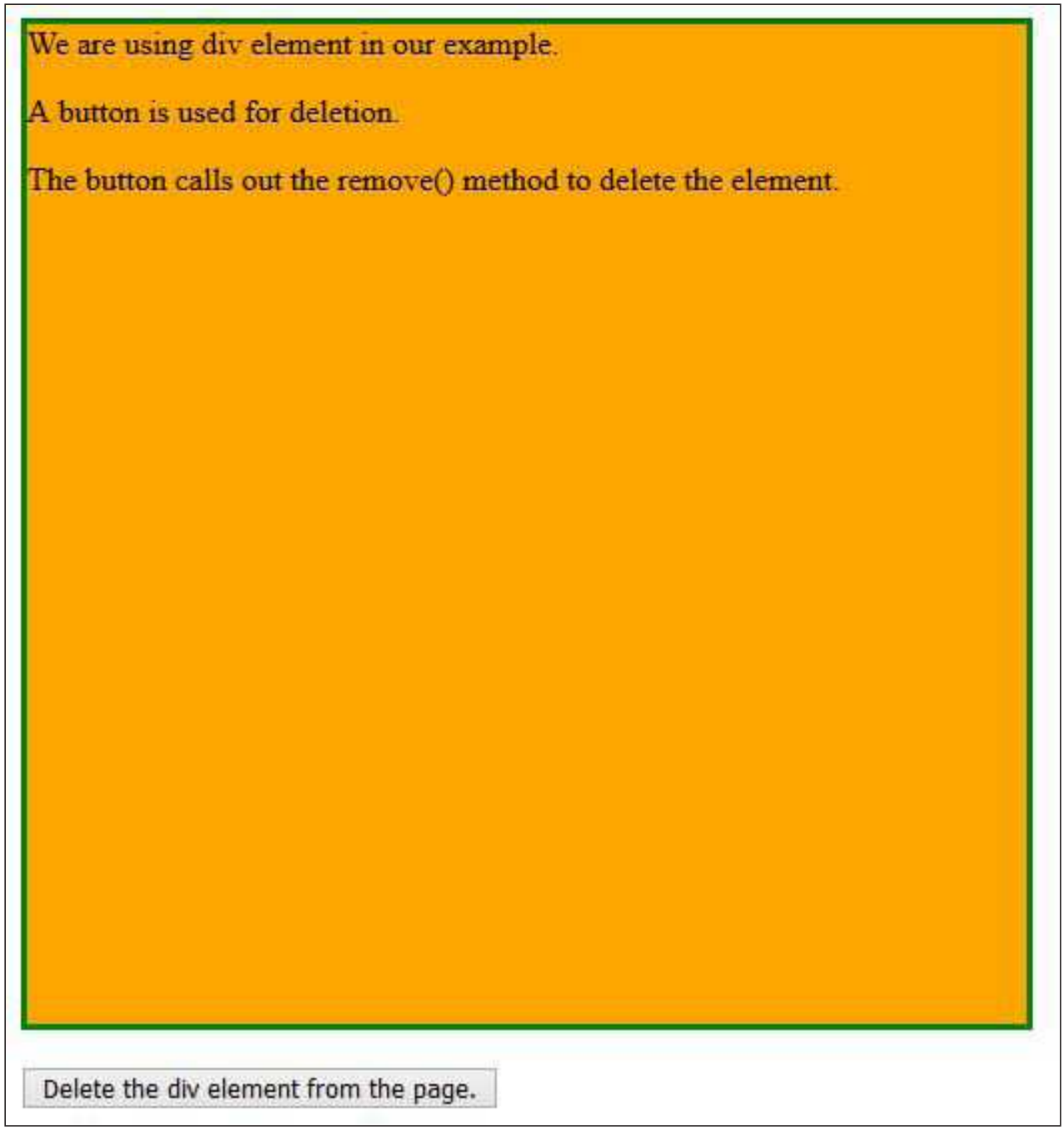
jQuery allows us to easily delete elements from the HTML. For this, two main methods used are `remove()` and `empty()`.

5.7.5.1 `remove()`

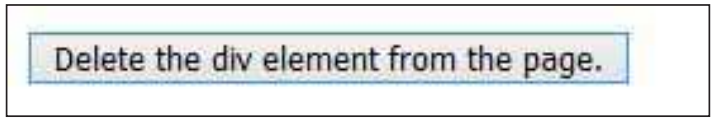
To delete any selected single element or set of elements, jQuery provides `remove()` method. It also deletes the child elements of those elements. In the following example, we designed a *div* element and called the `remove()` method with it.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("#d").remove();
     });
    });
    </script>
</head>
<body>
    <div id="d" style="height:500px;width:500px;border:3px solid green;
background-color:orange;">
    We are using div element in our example.
    <p>A button is used for deletion.</p>
    <p>The button calls out the remove() method to delete the element.</p>
    </div>
    <br>
    <button>Delete the div element from the page.</button>
</body>
</html>
```

This code shows the following result in a browser window which has orange background color and green border.



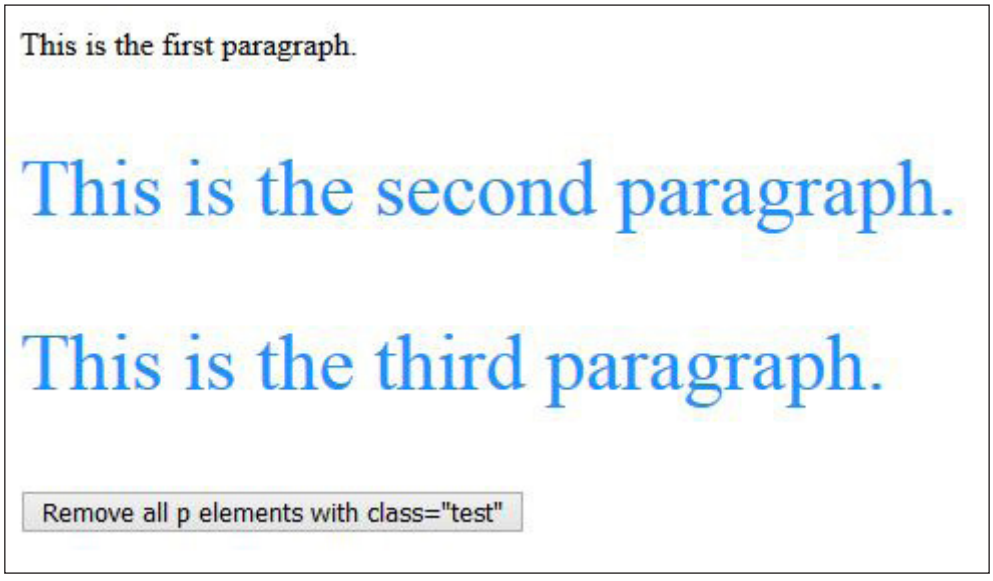
After the button is clicked, the following result will be displayed, where the entire div is deleted.



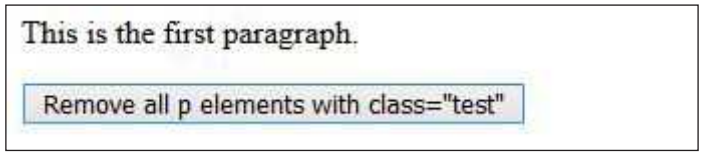
The `remove()` method can also take one parameter for filtering of elements. For instance, to make sure that all the paragraphs with "*b*" class are deleted, we can write the following.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("p").remove(".b");
     });
    });
    </script>
    <style>
    .b {
     color: dodgerblue;
     font-size: 40px;
    }
    </style>
</head>
<body>
    <p class ="a">This is the first paragraph.</p>
    <p class="b">This is the second paragraph.</p>
    <p class="b">This is the third paragraph.</p>
    <button>Remove all p elements with class="test"</button>
</body>
</html>
```

This code shows the following result in a browser window where the text of p element which got class b is shown in dodgerblue.



After the button is clicked, the following result will be displayed.

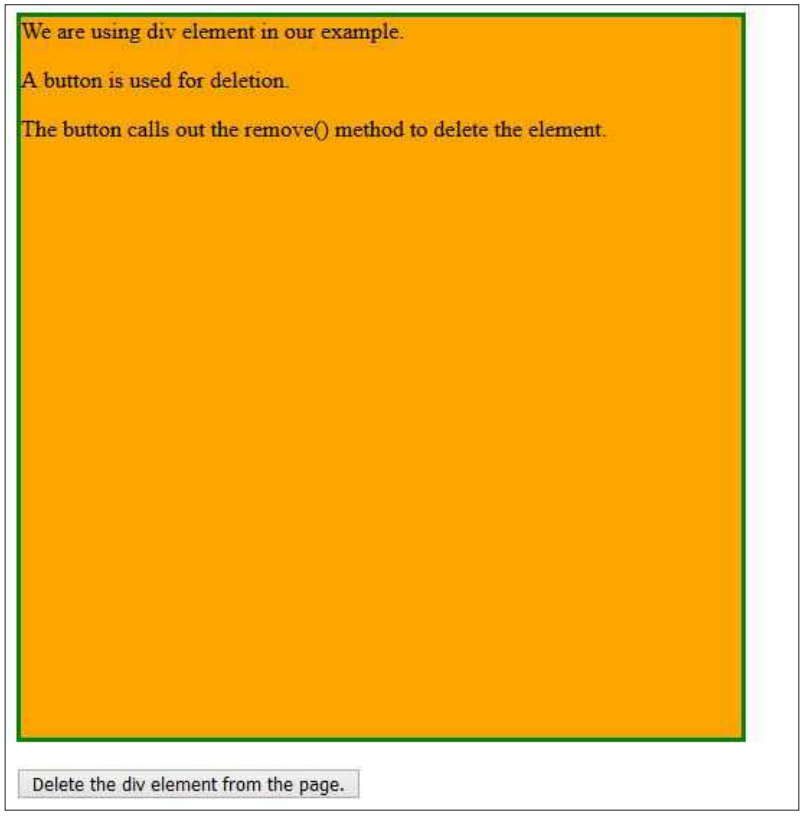


5.7.5.2 empty()

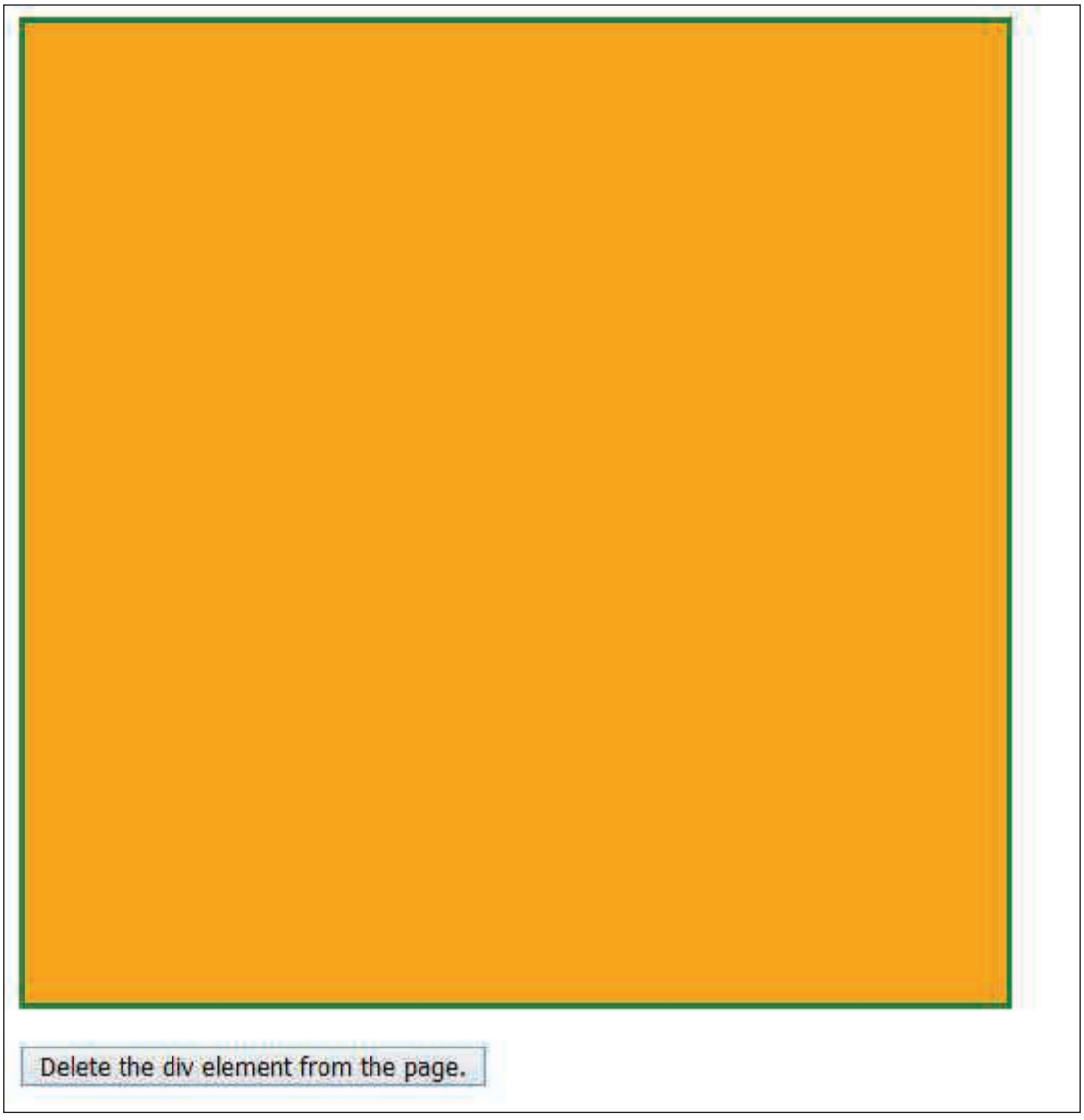
In order to delete an element's child elements, the `empty()` method is used. For instance, consider the following example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("#d").empty();
     });
    });
    </script>
</head>
<body>
    <div id="d" style="height:500px;width:500px;border:3px solid green;
background-color:orange;">
    We are using div element in our example.
    <p>A button is used for deletion.</p>
    <p>The button calls out the remove() method to delete the element.</p>
    </div>
    <br>
    <button>Delete the div element from the page.</button>
</body>
</html>
```

This code shows the following result in a browser window with orange background and green border.



After the button is clicked, the following result will be displayed, where the text disappears.



Flash Quiz Can jQuery connect with databases?

5.8 | jQuery with CSS

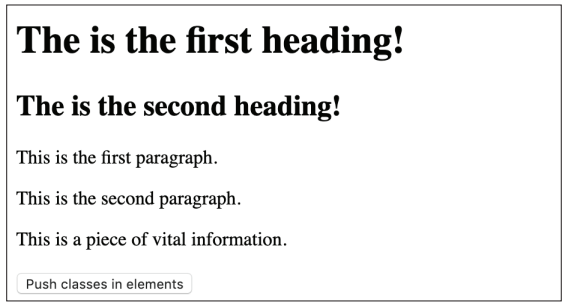
In addition to HTML, jQuery can also manipulate CSS elements. In this section, we will learn various jQuery methods to add, delete, or modify CSS properties.

5.8.1 `addClass()`

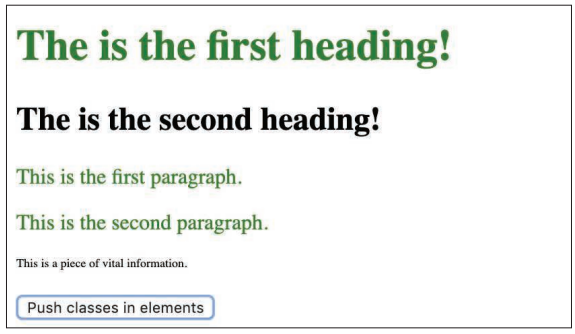
By using `addClass()`, CSS class attributes can be added in various HTML elements. You can pick a single or multiple elements and assign a class with it. For example, in the following example, we have assigned a CSS class "green" with our first heading and paragraph to change its color. Similarly, we have modified the size of our *div* element by using a "*size*" class.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("#b1").click(function(){
        $("h1, p").addClass("green");
        $("div").addClass("size");
     });
    });
    </script>
    <style>
    .size {
     font-size: xx-small;
    }
    .green {
     color: green;
    }
    </style>
</head>
<body>
    <h1>The is the first heading!</h1>
    <h2>The is the second heading!</h2>
    <p>This is the first paragraph.</p>
    <p>This is the second paragraph.</p>
    <div>This is a piece of vital information.</div><br>
    <button id ="b1">Push classes in elements</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the button click, the following result will be displayed, which turns the heading and p element content into green color and size of div element.



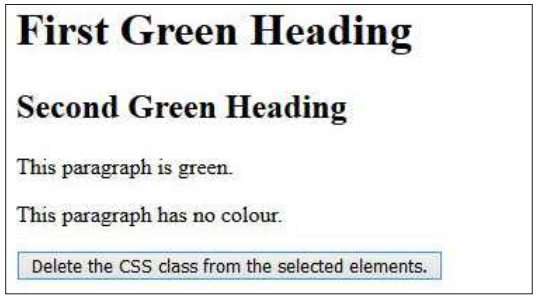
5.8.2 removeClass()

To delete any selected class attribute from multiple elements, you can use the `removeClass()` method. In the following example, we have eliminated the changes of CSS code.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("h1, h2, p").removeClass("green");
     });
    });
    </script>
    <style>
    .green {
     color: green;
    }
    </style>
</head>
<body>
    <h1 class="green">First Green Heading</h1>
    <h2 class="green">Second Green Heading</h2>
    <p class="green">This paragraph is green.</p>
    <p>This paragraph has no colour.</p>
    <button>Delete the CSS class from the selected elements.</button>
</body>
</html>
```

This code shows the following result in a browser window with h1, h2, and first p tag content in green.

After the button click, the following result will be displayed.

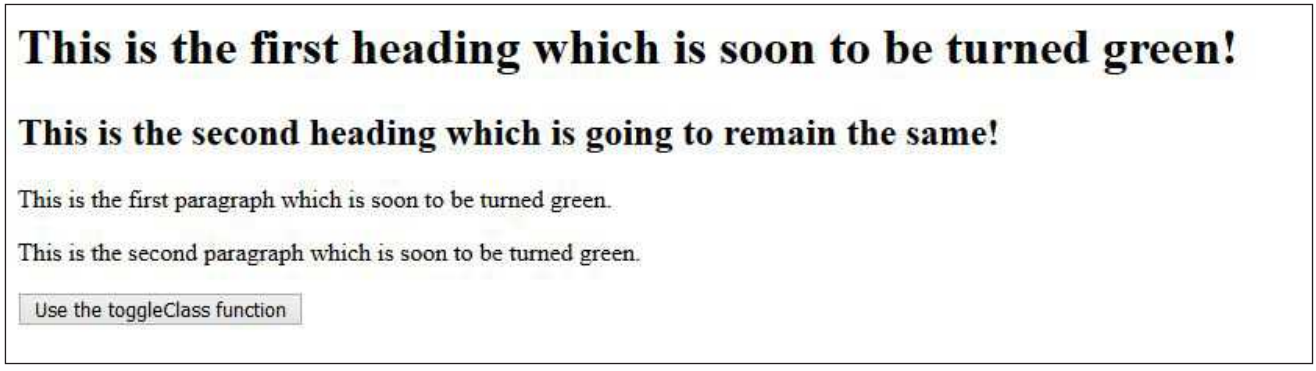


5.8.3 toggleClass()

By using the jQuery `toggleClass()` method, we can swap the functionalities of `addClass()` and `removeClass()` methods, that is, it adds and deletes classes from HTML's elements. In the following example, we have done exactly the same.

```
<!DOCTYPE html>
<html>
    <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("h1, p").toggleClass("green");
     });
    });
    </script>
    <style>
    .green{
     color: green;
    }
    </style>
</head>
<body>
    <h1>This is the first heading which is soon to be turned green!</h1>
    <h2>This is the second heading which is going to remain the same!</h2>
    <p>This is the first paragraph which is soon to be turned green.</p>
    <p>This is the second paragraph which is soon to be turned green.</p>
    <button>Use the toggleClass function</button>
</body>
</html>
```

This code shows the following result in the browser window.

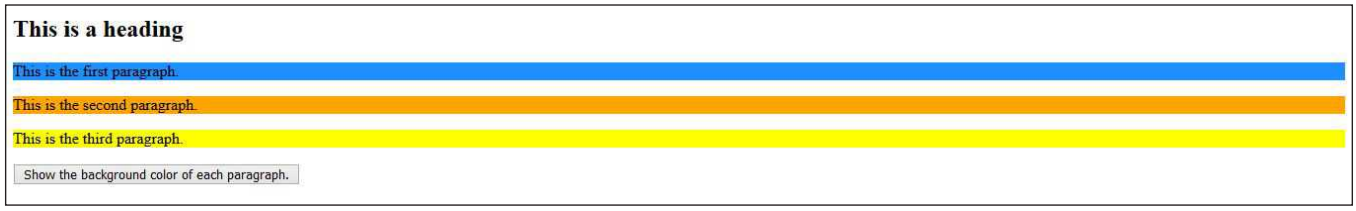


5.8.4 css()

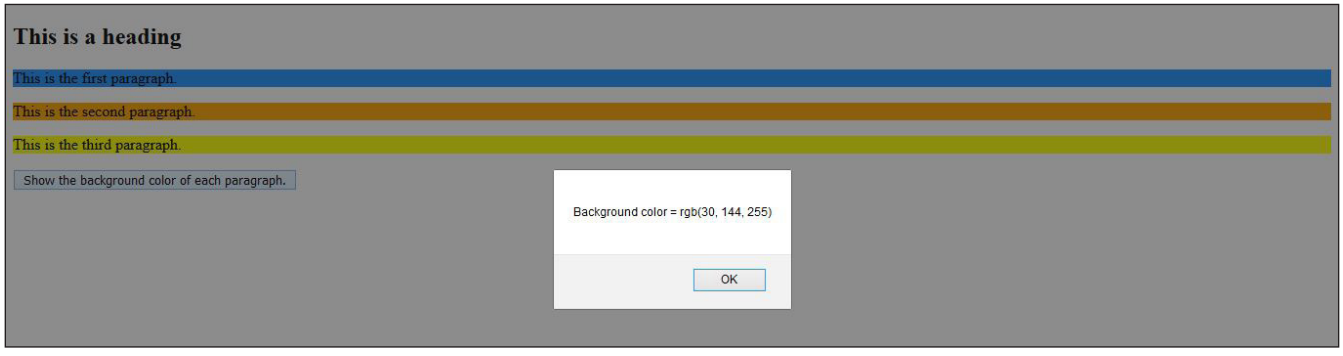
In order to set or return style properties for the specified elements, the `css()` method is used. In our example, we have assigned a separate color to each of our paragraph. An alert is used to display the CSS property of background color for each paragraph which shows the RGB values.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        alert("Background color = " + $("#p1").css("background-color"));
        alert("Background color = " + $("#p2").css("background-color"));
        alert("Background color = " + $("#p3").css("background-color"));
     });
    });
    </script>
</head>
<body>
    <h2>This is a heading</h2>
    <p id="p1" style="background-color:dodgerblue">This is the first paragraph.</p>
    <p id="p2" style="background-color:orange">This is the second paragraph.</p>
    <p id="p3" style="background-color:yellow">This is the third paragraph.</p>
    <button>Show the background color of each paragraph.</button>
</body>
</html>
```

This code shows the following result in the browser window.



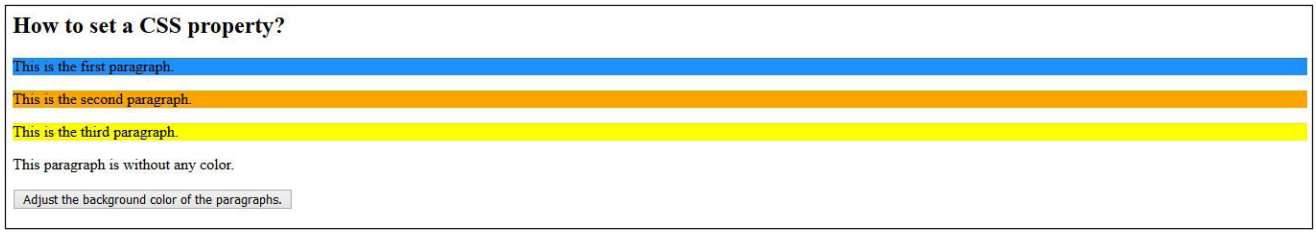
After the button click, the following result will be displayed.



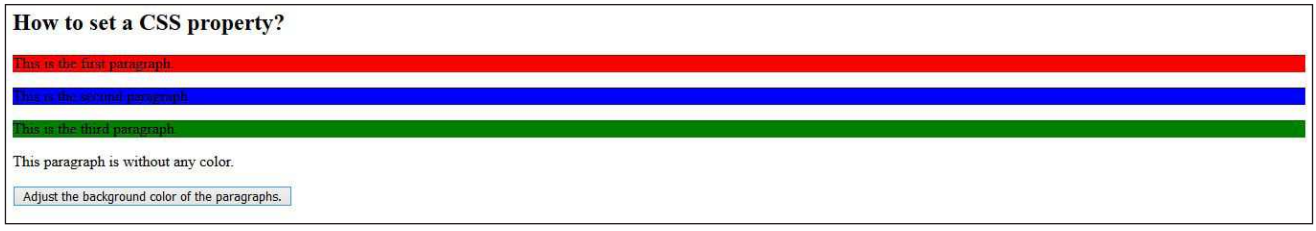
You can also use the CSS method to adjust an already set property in CSS. For instance, we had set colors for our paragraphs. By using the CSS method, we have modified them by accessing and changing the color property.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("button").click(function(){
        $("#p1").css("background-color", "red");
        $("#p2").css("background-color", "blue");
        $("#p3").css("background-color", "green");
     });
    });
    </script>
</head>
<body>
    <h2>How to set a CSS property? </h2>
    <p id="p1" style="background-color:dodgerblue">This is the first paragraph.</p>
    <p id="p2" style="background-color:orange">This is the second paragraph.</p>
    <p id="p3" style="background-color:yellow">This is the third paragraph.</p>
    <p>This paragraph is without any color.</p>
    <button>Adjust the background color of the paragraphs.</button>
</body>
</html>
```

This code shows the following result in the browser window.



After the button click, the following result will be displayed.



5.9 | Traversing

jQuery provides excellent traversing functionality. Traversing allows moving around HTML elements in accordance with their link to different elements. By beginning with one selection, you can go further and proceed with your filtering till you get to your intended element.

5.9.1 Ancestors

Ancestor refers to an element which may be a parent or grandparent of a child element. jQuery helps in traversal of the DOM tree by searching any element. There are three methods in jQuery to do this – `parent()`, `parents()`, and `parentsUntil()`.

Figure 5.1 shows an example DOM tree which starts with Document and has a root element named <html>. This is followed by two main elements <head> and <body>. <head> contains header related information like page title, meta data,

etc. And <body> contains all the elements which will be displayed on a browser window. Every element has various attributes and most of the elements may also contain text. See the following <button> element example which has attribute "type" and text which can be specified by the developer.

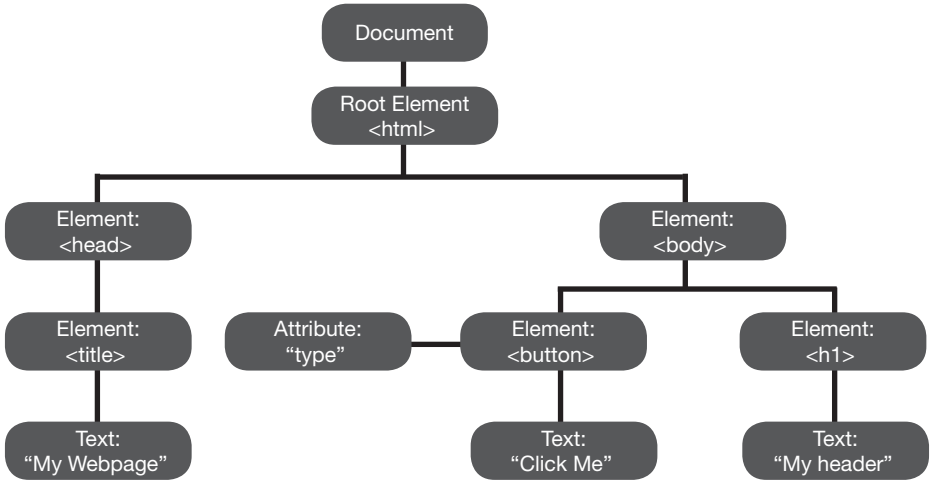


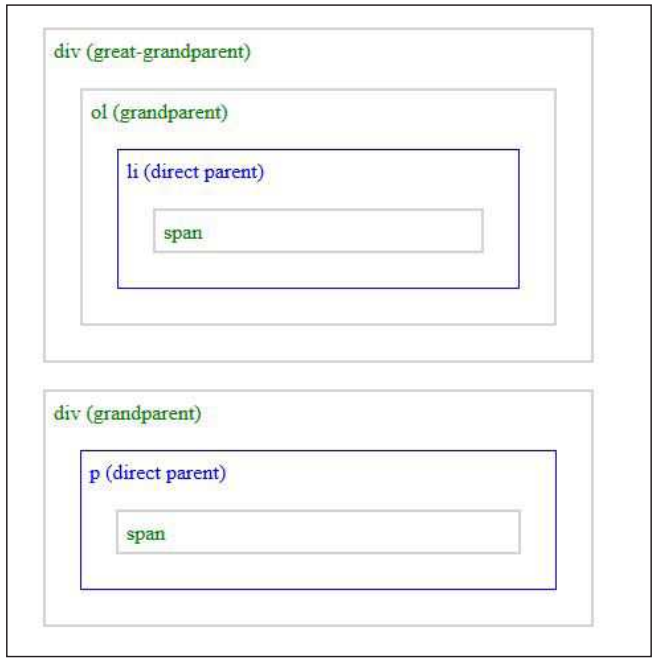
Figure 5.1 DOM tree.

5.9.1.1 `parent()`

To view the direct parent of a selected element, you can use the `parent()` method. It merely traverses one step up in the DOM. For example, the following code displays the direct parent for all the span elements.

```
<!DOCTYPE html>
<html>
<head>
   <style>
   .outermost * {
    display: block;
    border: 2px solid lightgrey;
    color: green;
    padding: 6px;
    margin: 20px;}
   </style>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
   <script>
   $(document).ready(function(){
    $("span").parent().css({"color": "blue", "border": "1px solid blue"});
   });
   </script>
</head>
<body>
   <div class="outermost">
    <div style="width:400px;">div (great-grandparent)
      <ol>ol (grandparent)
       <li>li (direct parent)
         <span>span</span>
       </li>
      </ol>
   </div>
    <div style="width:400px;">div (grandparent)
      <p>p (direct parent)
       <span>span</span>
      </p>
    </div>
   </div>
</body>
</html>
```

This code shows the following result in the browser window.

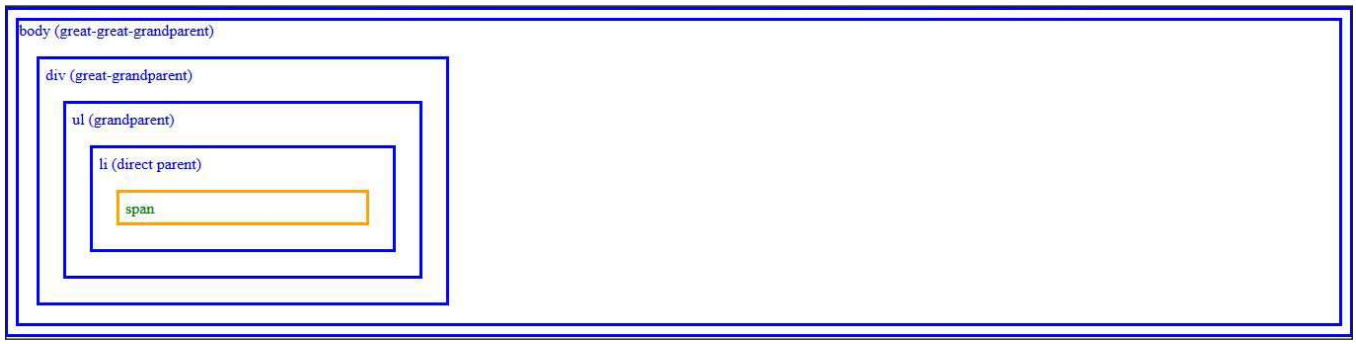


5.9.1.2 parents()

Likewise, the `parents()` method can be used to display all the ancestors from the direct parent to the outermost parent, that is, the root html tag.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .outermost * {
     display: block;
     border: 3px solid orange;
     color: green;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("span").parents().css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="outermost">body (great-great-grandparent)
     <div style="width:400px;">div (great-grandparent)
        <ul>ul (grandparent)
         <li>li (direct parent)
            <span>span</span>
         </li>
        </ul>
     </div>
</body>
</html>
```

This code shows the following result in the browser window.

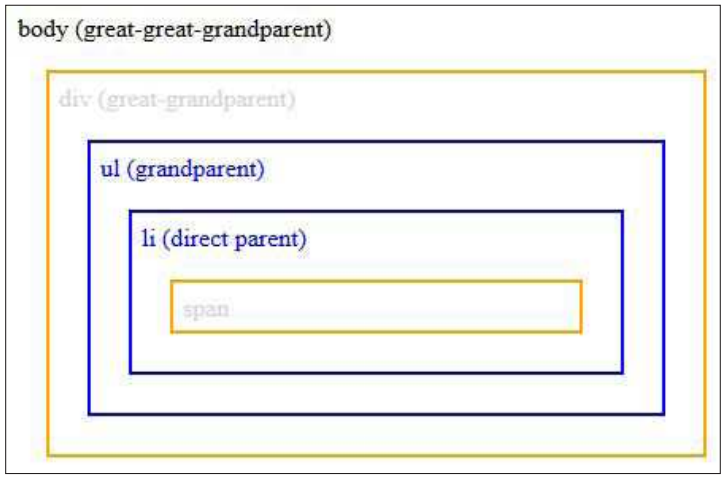


5.9.1.3 parentsUntil()

To search for all the ancestors till a specific point, you can use the `parentsUntil()` method. For instance, in the following example, you can get all parents starting from the span tag and ending at the *div* tag.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .outermost * {
     display: block;
     border: 2px solid orange;
     color: lightgrey;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("span").parentsUntil("div").css({"color": "blue", "border": "2px solid blue"});
    });
    </script>
</head>
<body class="outermost"> body (great-great-grandparent)
     <div style="width:400px;">div (great-grandparent)
        <ul>ul (grandparent)
         <li>li (direct parent)
            <span>span</span>
         </li>
        </ul>
     </div>
</body>
</html>
```

This code shows the following result in the browser window.



5.9.2 Descendants

As jQuery provides functionality to go up in the DOM tree, it also assists in going down in the DOM tree to select any descendant. A descendant can be any child or grandchild of an element. There are two main methods to find descendants:

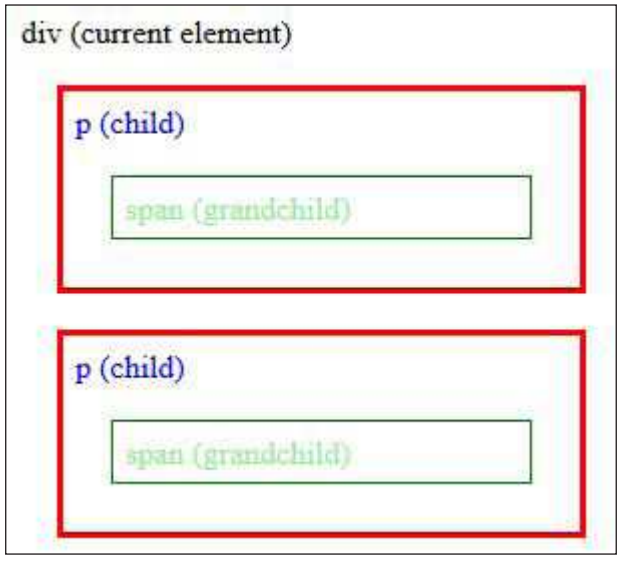
1. `children()`
2. `find()`

5.9.2.1 `children()`

To view the direct children of an element, you can use the `children()` method. It merely goes down to show a single level in the DOM. For instance, to check the direct child of the *div* tag, we have used the `children()` method by a blue-color representation.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .innermost * {
     display: block;
     border: 1px solid green;
     color: lightgreen;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("div").children().css({"color": "blue", "border": "3px solid red"});
    });
    </script>
</head>
<body>
    <div class="innermost" style="width:300px;">div (current element)
     <p>p (child)
       <span>span (grandchild)</span>
     </p>
     <p>p (child)
       <span>span (grandchild)</span>
     </p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.



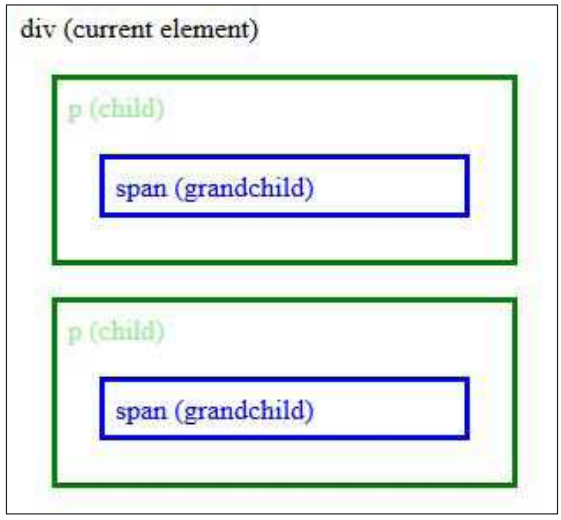
AQ1

5.9.2.2 find()

To view all the existing children of a specified element, the `find()` method is used. Consider the following example in which we will find all the span element which are child to div element and css color properties.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .innermost * {
     display: block;
     border: 3px solid green;
     color: lightgreen;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("div").find("span").css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body>
    <div class="innermost" style="width:300px;">div (current element)
     <p>p (child)
        <span>span (grandchild)</span>
     </p>
     <p>p (child)
        <span>span (grandchild)</span>
     </p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.



5.9.3 Siblings

Siblings elements are the ones who share the same parent. jQuery allows to traverse horizontally to search an element's siblings. To check a sibling, there are multiple methods such as `siblings()`, `next()`, `nextAll()`, `nextUntil()`, `prev()`, `prevAll()`, and `prevUntil()`.

5.9.3.1 `siblings()`

To check the siblings of a specified element, you can use the `siblings()` method. For example, to check the siblings of the second heading <*h2*>, we have used the `siblings()` method.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid yellow;
     color: orange;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h2").siblings().css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p</p>
     <span>span</span>
     <h2>h2</h2>
     <h3>h3</h3>
     <p>p</p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.3.2 next()

To view the next sibling of a specified element, you can use the `next()` method. For example, to check the next sibling of *h2* and *h3*, we have written the following code:

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid green;
     color: green;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h2").next().css({"color": "blue", "border": "3px solid blue"});
    });
    $(document).ready(function(){
     $("h3").next().css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p</p>
     <span>span</span>
     <h2>h2</h2>
     <h3>h3</h3>
     <p>p</p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.3.3 nextAll()

To view all the siblings which are next to a selected element, you can use the `nextAll()` method. For example, to view h2's next siblings you can write the following code:

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid green;
     color: green;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h2").nextAll().css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p1</p>
     <p>p2</p>
     <span>span</span>
     <h2>h2</h2>
     <h3>h3</h3>
     <span>span</span>
     <p>p3</p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.3.4 `nextUntil()`

To view the next siblings of element till a given point is reached, you can use the `nextUntil()` method. In our example, we started searching for the next siblings of *h1* till *h5*.

```
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid green;
     color: lightgrey;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h1").nextUntil("h5").css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p</p>
     <span>span</span>
     <h1>h1</h1>
     <h2>h2</h2>
     <h3>h3</h3>
     <h4>h4</h4>
     <h5>h5</h5>
     <h6>h6</h6>
     <h6>h7</h6>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.3.5 prev()

You can also check the previous elements of a sibling by using the `prev()` method. Consider the following example where we check the immediate previous sibling of *h5*:

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid green;
     color: lightgrey;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h5").prev().css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p</p>
     <span>span</span>
     <h1>h1</h1>
     <h2>h2</h2>
     <h3>h3</h3>
     <h4>h4</h4>
     <h5>h5</h5>
     <h6>h6</h6>
     <h6>h7</h6>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.3.6 `prevAll()`

Similarly, to check all the previous siblings of *h5*, you can use the `prevAll()` method.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid green;
     color: lightgrey;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h5").prevAll().css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p</p>
     <span>span</span>
     <h1>h1</h1>
     <h2>h2</h2>
     <h3>h3</h3>
     <h4>h4</h4>
     <h5>h5</h5>
     <h6>h6</h6>
     <h6>h7</h6>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.3.7 prevUntil()

Additionally, to check the previous siblings of *h5* to a certain extent you can use the `prevUntil()` method.

```
<!DOCTYPE html>
<html>
<head>
    <style>
    .s * {
     display: block;
     border: 3px solid green;
     color: lightgrey;
     padding: 6px;
     margin: 18px;
    }
    </style>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("h5").prevUntil("span").css({"color": "blue", "border": "3px solid blue"});
    });
    </script>
</head>
<body class="s">
    <div>div (parent)
     <p>p</p>
     <span>span</span>
     <h1>h1</h1>
     <h2>h2</h2>
     <h3>h3</h3>
     <h4>h4</h4>
     <h5>h5</h5>
     <h6>h6</h6>
     <h6>h7</h6>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

5.9.4 Filtering

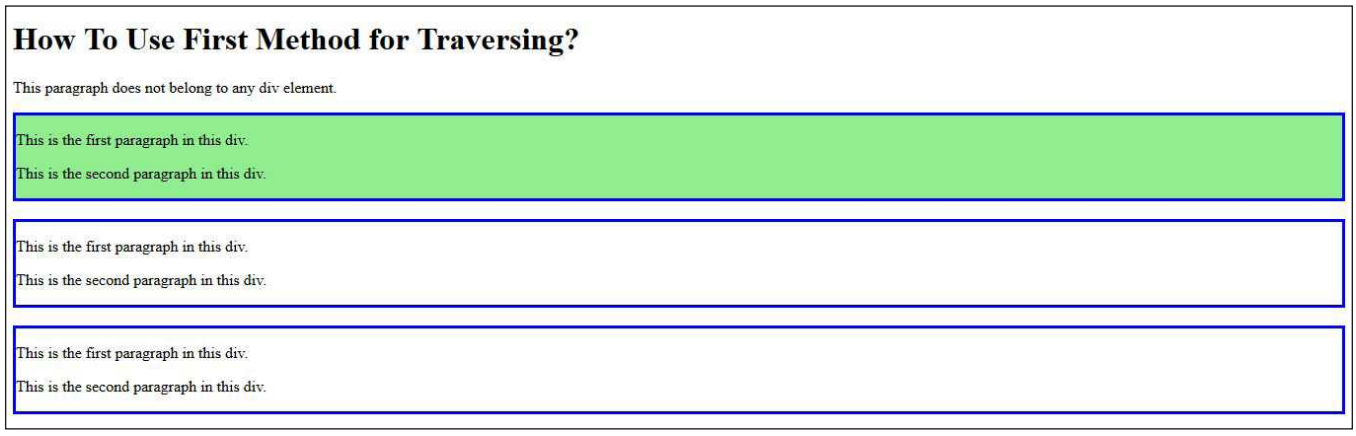
You can also traverse through filtering.

5.9.4.1 `first()`

To view the beginning element of a defined element, you can use the `first()` method. In this example, we have selected the beginning <*div*> element.

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("div").first().css("background-color", "lightgreen");
    });
    </script>
</head>
<body>
    <h1>How To Use First Method for Traversing?</h1>
    <p>This paragraph does not belong to any div element.</p>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
    <br>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
    <br>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

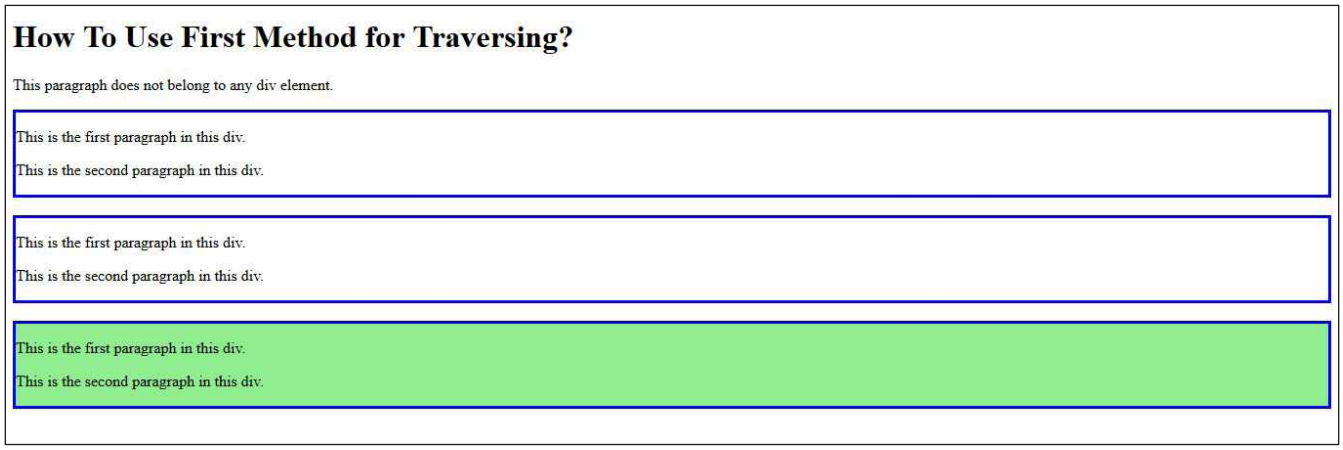


5.9.4.2 `last()`

Similarly, by using the `last()` method you can check the ending element of the selected elements. For example,

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("div").last().css("background-color", "lightgreen");
    });
    </script>
</head>
<body>
    <h1>How To Use First Method for Traversing?</h1>
    <p>This paragraph does not belong to any div element.</p>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
    <br>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
    <br>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.

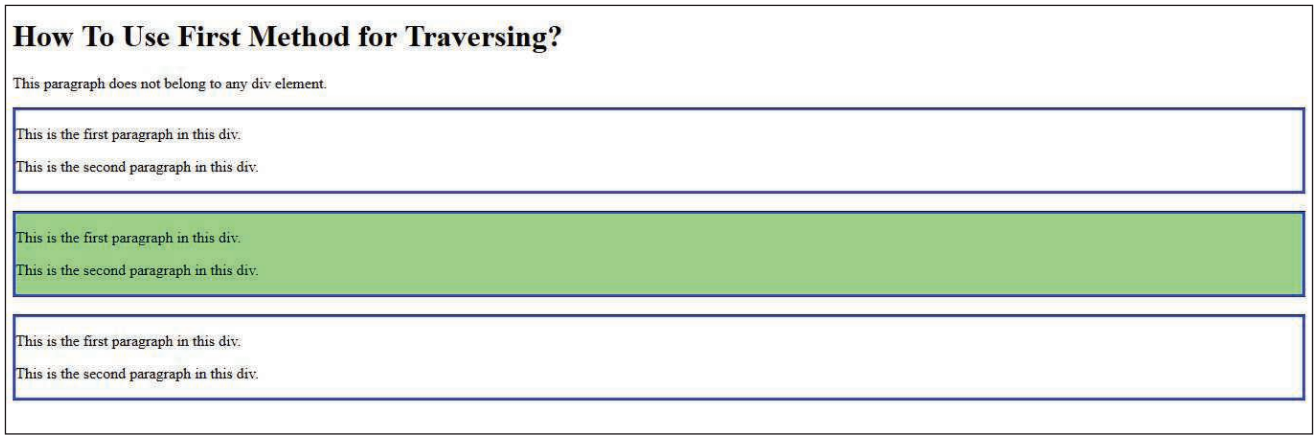


5.9.4.3 eq()

To select a specific element, you can use the `eq()` method. It takes the index number as an argument for the defined elements. The index begins with 0. Consider the following example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
    <script>
    $(document).ready(function(){
     $("div").eq(1).css("background-color", "lightgreen");
    });
    </script>
</head>
<body>
    <h1>How To Use First Method for Traversing?</h1>
    <p>This paragraph does not belong to any div element.</p>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
    <br>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
    <br>
    <div style="border: 3px solid blue;">
     <p>This is the first paragraph in this div.</p>
     <p>This is the second paragraph in this div.</p>
    </div>
</body>
</html>
```

This code shows the following result in the browser window.



Flash Quiz Can you extend current jQuery functionality? Hint: Creating a plugin.

Summary

jQuery is one of the popular frameworks for web application development. It greatly simplifies the use of JavaScript and provides various functionalities that ease the front-end development. Since the popularity of jQuery has increased significantly, there are a lot of plugins available which enhance the front-end processing capabilities.

In this chapter, we have learned the following concepts:

1. jQuery and its use to control HTML.
2. Various uses of jQuery for HTML tags.
3. jQuery selectors to manipulate content within elements.
4. Access parent elements and traverse between DOM elements to find first, last, etc.

In Chapter 6, we will learn about Bootstrap and how it is useful to develop a mobile-friendly application. It offers functionalities that help to design a slick and modern webpage, which will render properly on all the available screen sizes.

Multiple-Choice Questions

1. Which one of the following jQuery methods prevents the browser from executing the default action?
   - **(a)** `StopImmediatePropagation()`
   - **(b)** `StopPropagation()`
   - **(c)** `preventDefault()`
   - **(d)** None of the above
2. We can pass an anonymous function as an argument to another function.
   - **(a)** True
   - **(b)** False
3. Which one of the built-in methods in JavaScript sorts the elements of an array?
   - **(a)** `Sort()`
   - **(b)** `Order()`
   - **(c)** `changeOrder(order)`
   - **(d)** None of the above
4. Which of the given jQuery methods removes all child nodes from the set of matched elements?
   - **(a)** `delete()`
   - **(b)** `empty()`
   - **(c)** `remove(expr)`
   - **(d)** None of the above
5. Which one of the given jQuery methods filters out elements from a set of matched elements?
   - **(a)** `setFilter (selector)`
   - **(b)** `filter (selector)`
   - **(c)** `putFilter(selector)`
   - **(d)** None of the above

Review Questions

1. How can you select a particular tag to perform an action on it?
2. How can you select multiple tags at once to perform a common action on them?
3. How can you remove an HTML element on a button click?
4. How can you find the last element of a particular type of element like ?
5. How can you add style to the first element of a particular type of element like <a>?
6. How can you find parent of an element in nested elements case?
7. How do you add jQuery library to a page?
8. Can you change class of an element on a button click?
9. Can you add a class to an element on a button click?
10. How can you read value from a text box using jQuery?
11. Can you use jQuery and JavaScript code together on a page?
12. Is jQuery operating system dependent?
13. Can you have nested "document.ready" functions?
14. Do you need to install anything on the server to add jQuery code?

Exercises

1. Create a page which contains a table and add button which will add rows to this table.
2. Change color of a text on button click.
3. Show a popup box without using JavaScript alert.
4. Write code to fetch data from Wikipedia by using ajax function.
5. Add a text box and a button. Write code to remove all the elements which ID matches with the text entered in the text box upon clicking on the button.

Project Idea

Take the dictionary project idea from the previous chapters. Create a page that allows adding a word to the word list. This page should provide a text box and allow users to enter a word that they want to add to the dictionary list. It should also have 3 text areas where they can add example sentences, synonyms, and antonyms of the given word. In the end, add a button that will add the word and its example, synonyms, and antonyms to the word list table upon clicking on it.

Recommended Readings

1. Bear Bibeault, Yehuda Katz, Aurelio De Rosa, Dave Methvin, John Resig. 2016. *jQuery In Action, Third Edition, (MANNING)*. Dreamtech Press: New Delhi
2. Jon Raasch, Graham Murray, Vadim Ogievetsky, Joseph Lowery. 2015. *Javascript and jQuery for Data Analysis and Visualization*. Wiley: New Jersey
3. Richard York. 2015. *Web Development with jQuery*. Wrox: Birmingham
4. jQuery – https://jquery.com/
5. W3School – https://www.w3schools.com/jquery/
6. jQuery UI – https://jqueryui.com/